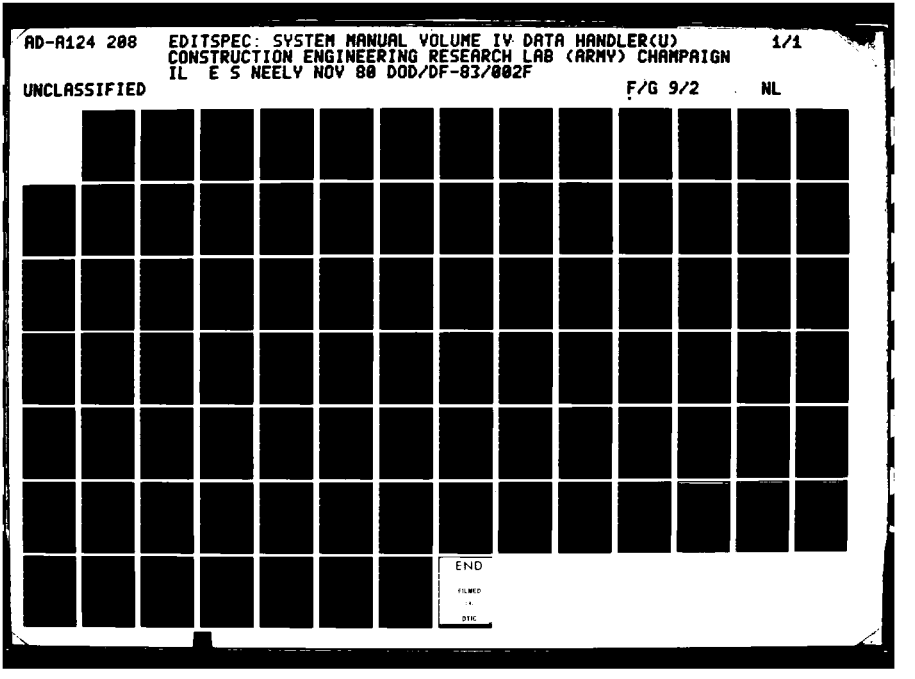
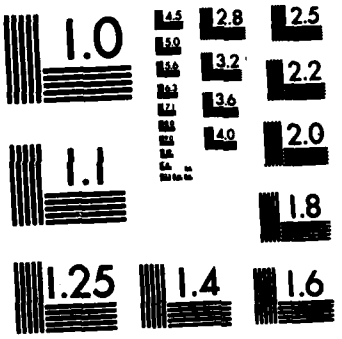

AD-A124 208
EDITSPEC: SYSTEM MANUAL VOLUME IV DATA HANDLER(U)
CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAIGN
IL E S NEELY NOV 80 DOD/DF-83/002F
1/1
UNCLASSIFIED
F/G 9/2
NL
END
FILMED
DTIC



Computer-Based Specifications

EDITSPEC SYSTEM MANUAL

VOLUME FOUR -- DATA HANDLER

by

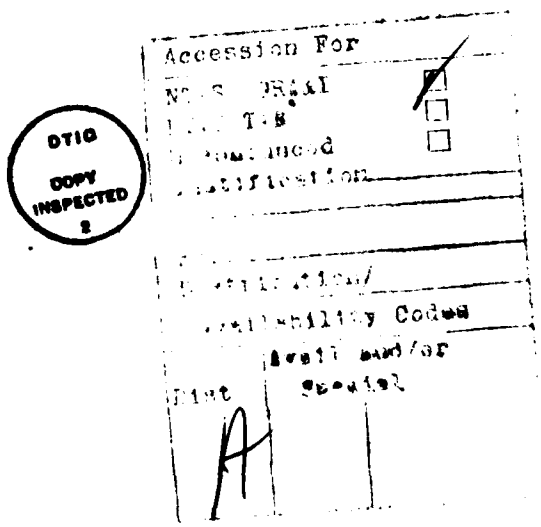
Edgar S. Neely Jr.

Final Technical Report

November 1980

PREFACE TO ALL EDITSPEC SYSTEM REPORTS

The purpose of the EDITSPEC system reports is to provide complete documentation to all personnel that must be involved with the EDITSPEC system. Such personnel include managers, specification writers, typists, computer systems analysts, and computer programmers. Each personnel group requires different documentation. The reports required and the order of reading are shown in the table below.



ABSTRACT

This report provides computer programmers with documentation for a machine-independent scientific and engineering data-handler. The report discusses design concepts, subroutine functions, application commands, error messages, program conversion, test routines, and procedures for changing the data-handler. The data-handler has been designed to minimize the resources required for its conversion and operation.

FOREWORD

This investigation was performed for the Directorate of Military Construction, Office of the Chief of Engineers (OCE), under Project 4A762731AT41, "Design, Construction, and Operation and Maintenance Technology for Military Facilities"; Task T1, Development Work Unit 009, "Computer-Based Specifications." The applicable QCR is 1.10.001. The OCE Technical Monitor was William Darnell.

The basic computer programs were written by Multi-Systems Inc., Cambridge, Mass., under contract DACA 23-75-C-0003, and modified by Hans Wegener and Jayant Krishnaswamy of CERL.

The study was performed by the Management Systems Branch (Dr. O. E. Rood, Jr., Chief), Facility Acquisition and Construction Division (Mr. E. A. Lotz, Chief), U.S. Army Construction Engineering Research Laboratory (CERL).

COL Louis J. Circeo is Commander and Director of CERL, and Dr. L. R. Shaffer is is Technical Director.

LIST OF REVISIONS TO THIS DOCUMENT

Revision Number	Date Revised	Reason for Revisions	Pages Revised

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. DOD/DF-83/002f	2.	3. Recipient's Accession No. AD-A124 208
4. Title and Subtitle: EDITSPEC: System Manual, Volume IV: Data Handler			5. Report Date: November 1980
			6.
7. Author(s): Edgar S. Neely, Jr.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address: Department of the Army, Construction Engineering Research Laboratory, P.O. Box 4005, Champaign, IL 61820			10. Project/Task/Work Unit No. 4A762731AT41/T1/009; 11. Contract(C) or Grant(G) No. (C) (G)
12. Sponsoring Organization Name and Address: (same)			13. Type of Report & Period Covered; 14.

15. Supplementary Notes

For magnetic tapes, see

16. Abstract (Limit: 200 words)

The EDITSPEC System is an automated system designed to produce construction specifications from Corps of Engineers Guide Specifications. The System uses one central computer and a communications network to provide remote terminal access by Corps offices, nationwide to a central data base.

This report provides computer programmers with documentation for a machine-independent scientific and engineering data-handler. The report discusses design concepts, subroutine functions, application commands, error messages, program conversion, test routines, and procedures for changing the data-handler. The data-handler has been designed to minimize the resources required for its conversion and operation.

17. Document Analysis a. Descriptors

Construction Specifications
Guide Specifications
Military Construction

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement:	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

EDITSPEC SYSTEMS MANUAL
VOLUME IV: DATA HANDLER

1 DESIGN CONCEPTS

Introduction

This data-handler is a feasible method by which a programmer can create data structures on disk without knowing record sizes, data locations, etc. This direct access system provides in-core data handling so that physical disk accesses are minimized.

The programs consist of several data storage and retrieval functions and a variety of other utilities which create and delete files, inform the user of file status, and establish or remove access restrictions. The programs are written mostly in ANSI FORTRAN.

Definitions

DATA SET: a named, independently accessible collection of data on a direct access storage device, created and maintained by the host operating system; called a file by some hardware manufacturers.

PHYSICAL RECORD: a physical portion of a data set usually of fixed length, which is accessible by means of one input/output operation the computing system.

FILE: a named, logical subdivision of a data set.

LOGICAL RECORD: a logical portion of a file of variable length, which is accessible through one application of the data-handler; usually referred to simply as "record."

In essence, the data-handler maintains a logical structure of named files composed of variable-length, randomly accessible records by manipulating physical records in one or more data sets.

APPLICATION PROGRAM: The set of subroutine that are using the DH for data base management.

Data Storage Modes

Two storage modes control how data is stored in the physical records on disk: CHAIN and PACK.

In CHAIN mode, the data-handll store with each logical record the record identifiers of the preceding and following logical records in the file. The logical records must therefore remain in the order in which they are written. Chain pointers are available to the programmer who is reading or updating logical records. The pointers are always updated whenever a logical record is added or lengthened.

In PACK mode, the data-handler routines will fill physical records to capacity and, when necessary, segment logical records which span two or more physical records.

CHAIN and PACK modes may be specified independently so that four combinations of options are available:

0 - neither CHAIN nor PACK
1 - CHAIN only
2 - PACK only
3 - CHAIN and PACK

The DH recognizes only the first definition of this option. Any redefinitions are ignored by the DH.

Data Management Implementation

The "data handler" (DH) is a general-purpose, direct-access data management system. The DH appears to an application program as a set of subroutines. The DH is written largely in ANS FORTRAN. The DH is designed as an interface between an application program and FORTRAN direct-access input/output (I/O) facilities (Figure 1). This provides a two-fold advantage. Direct-access I/O can be performed (1) at a logical level and (2) in a machine-independent manner.

The DH deals with only direct-access data sets (Figure 2). The contents of a dataset must be understood from three different view points:

(1) host system

(2) data handler

(3) application program

a. Host System. Data sets consist of physical records.

b. Data Handler. Data sets consist of files. Files consist of physical records. Physical records may also be referred to as

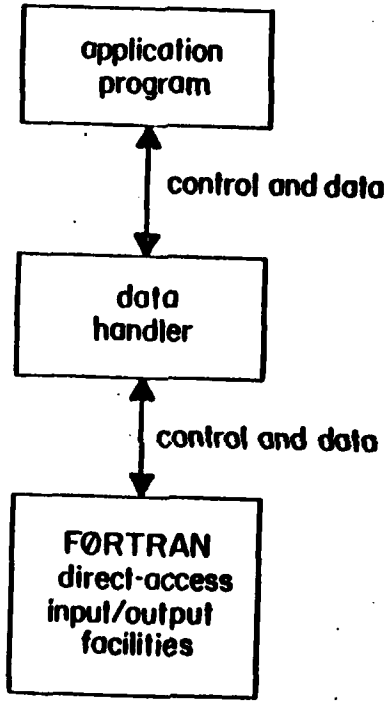


Figure I. Data Handler As Interface

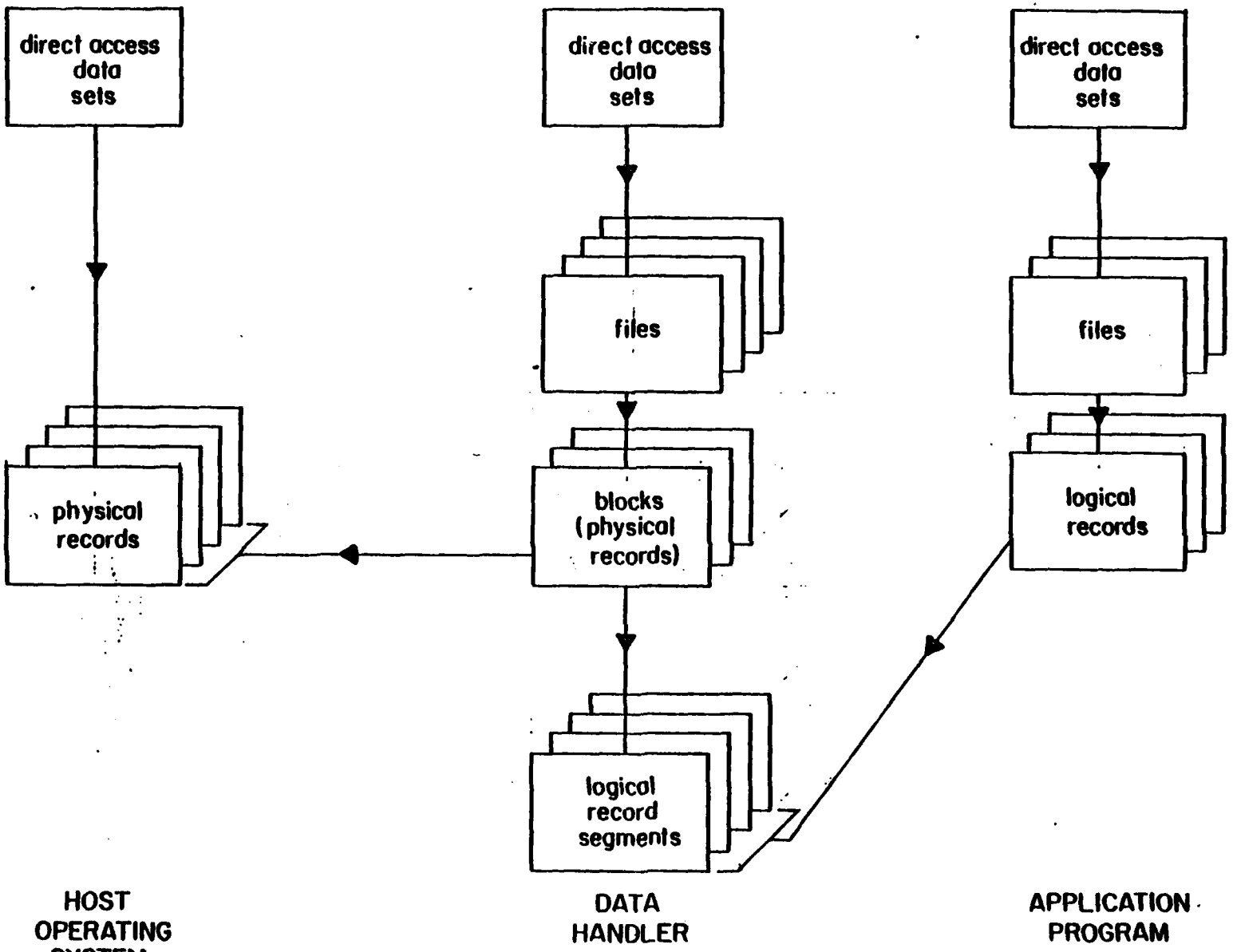


Figure 2. Direct Access Data Set Structure

blocks. Blocks (physical records) are composed of logical record segments.

c. Application Program. Data sets consist of files. Files consist of logical records.

Several cross-relationships should be noted:

(1) The logical record of the application program may actually be stored within one or more data handler logical record segments.

(2) A data handler block may be written as one or more operating system physical records.

Program Variable Naming Conventions

The names used for variables within the data handler follow the following logical order:

(1) All variable names are five letters in length.

(2) The first letter options are:

a. K for contents of

b. I for index to

c. N for number of

(3) The second and third letters options are:

a. PR for physical record

b. LR for logical record

c. DS for data set

d. OF for open file

e. BT for block table

(4) The fourth letter options are:

a. H for header

(5) The fifth letter options are:

a. S for status

b. P for previous

c. F for following

d. N for number

Data Set Structures

The first two blocks of a direct-access data set maintained by the DH contain three independent, special-purpose LR segments (Figure 3). These blocks do not belong to any file, and the segments are known as the data set header (DSHED), the file directory (FD), and the free space table (FST). The DSHED and FD are contained in the first block, while the FST is contained in the second.

The DSHED contains various information about the data set and host computer system (Figure 4). See the discussion of data set table entry below for the meanings of KDSSU, KDSPR, KDSBI, KDSFS, and KDSFD. See Table 1 for the meanings of NBSGN, NBBLK, DHFLG, ØPSYS(2), and DHVER. The time is given in the form hh:mm:ss, and the date is given left-justified and blank-padded in the form dd-mmm-yy; they are the time and date when the data set was initialized by the DH. The rest of the elements in the DSHED are the same as identically-named elements in blank CØMMØN.

The FD contains an entry for each file that is contained in the data set. An FD entry is identical to an open file table entry. FD entries are numbered consecutively beginning with one.

The FST is a bit map; each bit indicates whether (on) or not (off) the block corresponding to that bit has been allocated to a file in the data set. Both the FD and FST are extended when necessary to additional LR segments.

The remaining blocks are used to store data. The data is organized into dynamic collections of blocks known as files. One block can belong to only one file.

Block Structures

Blocks within a dataset are sequentially numbered starting with one. A block can belong to only one file. The block number is the sequential number assigned to a block. A block is composed of three sections.

The first three SU's in each block are known as the physical record (PR) header (figures). The first SU contains the contents of the physical record header status (KPRHS). The high-order (left-hand) half of KPRHS is the number of logical record segments in the block, while the low-order half is the number of unused SU's at the end of the block. The second SU KPRHP is the block number of the block that was previously written for the file to which this block belongs; it is zero if this block is the first one in the file. The

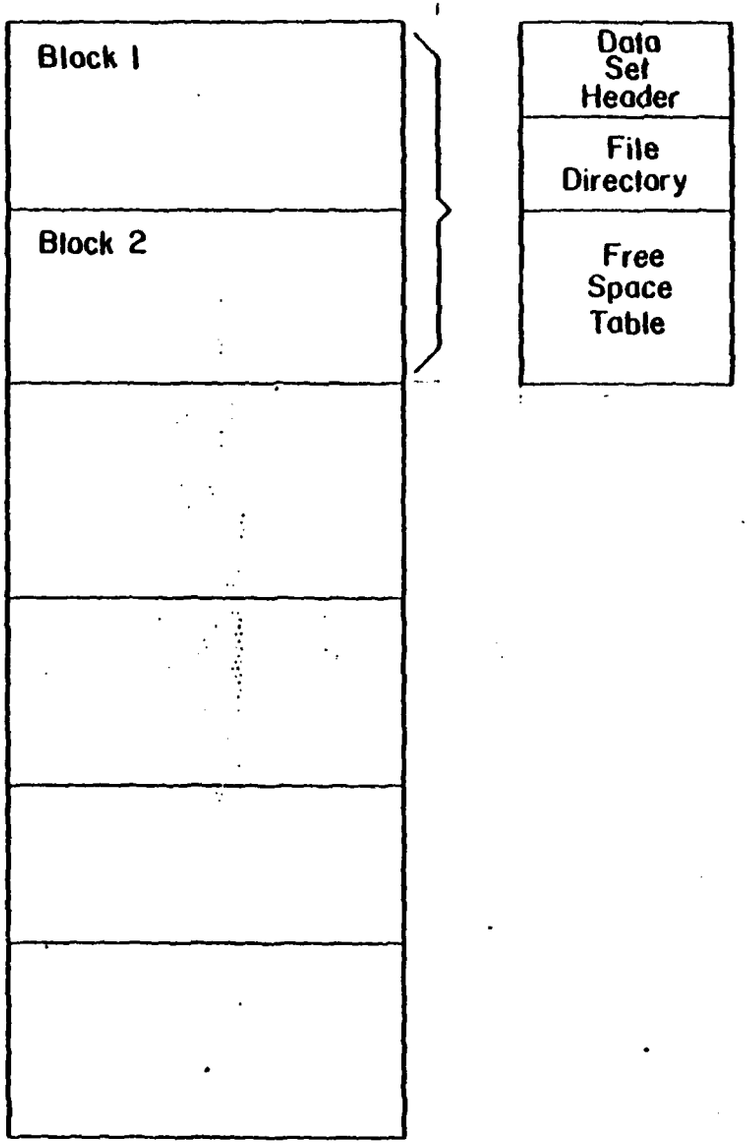


Figure 3. Data Set Layout

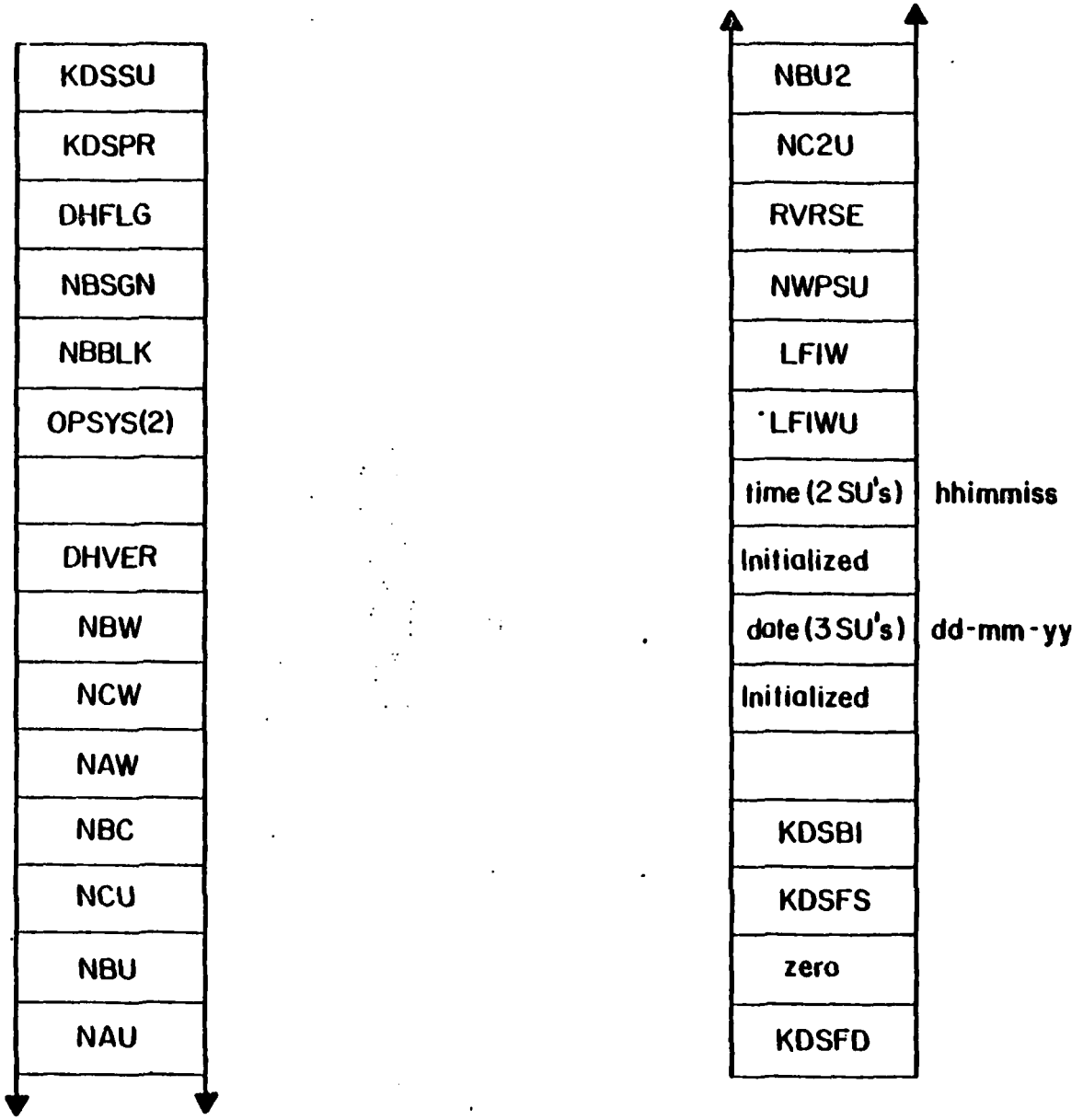


Figure 4. Data Set Header

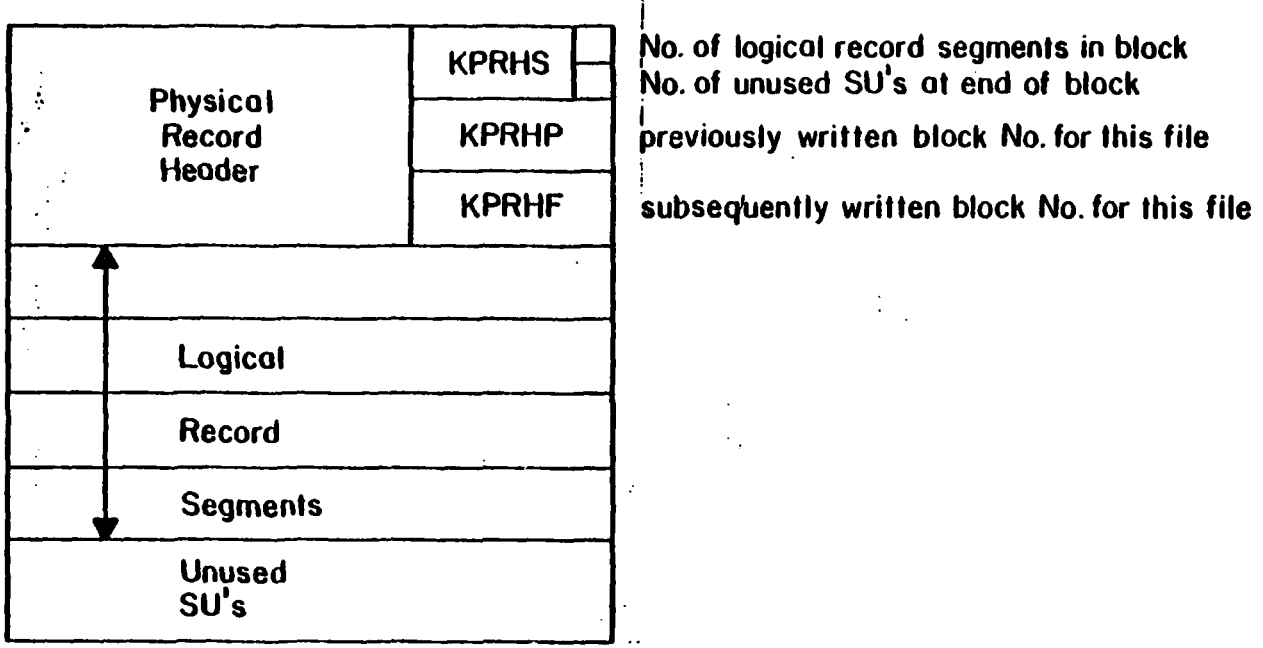


Figure 5. Block Layout

third SU KPRHF is the block number of the block that was subsequently written or is next to be added to this file. Hence, the blocks that constitute a file are chained together in both directions.

The application programs logical record may be divided by the DH into segments for storage. These segments are known as logical record segments. The block will contain one or more logical record (LR) segments. There are three reasons why an LR may be represented by more than one segment: (1) the LR is too long to fit in one block; (2) the LR is too long to fit into the unused portion of any block belonging to the file for which this LR is written, and the pack option is in effect for this file; (3) the LR cannot be contiguously extended due to lack of space in the block.

The remaining portion of the block contains unused SU's.

Logical Record Segment Structure

Associated with each LR segment is an SU quantity known as the LR segment identifier (ID). The high-order quarter of this quantity is the segment number, which is the position of the segment within the block, relative to the beginning of the block; segments are numbered beginning with zero. The low-order three-quarters of the LR segment ID constitute the block number. If the LR segment is the first or only segment of the LR, then the LR segment ID is often called simply the LR ID.

The first several SU's in each LR segment are known as the LR segment header (Figure 6). The extreme high-order portion of KLRHS contains three status bits known as the continue, segment, and delete bits, whose positions are given by the CØMMØN variables NBCØN, NBSEG, and NBDEL, respectively (Table 1). Bits within an SU are numbered from right to left beginning with zero. The continue bit indicates whether (on) or not (off) this LR segment is continued to a subsequent segment. The segment bit indicates whether (on) or not (off) this segment is continued from a previous segment. The delete bit indicates whether or not this LR has been deleted. The four low-order bits in the high-order half constitute the number of free characters in the last SU. The low-order half of KLRHS is the number of SU's in the LR (segment) including the header. The segments that represent an LR are chained together in both directions, but the LR's that constitute a file are chained only if the chain option is in effect for the file (Figure 7).

KLRHP contains the following:

No Chain Option

(1) If this LRS is the first one in the file, zero.

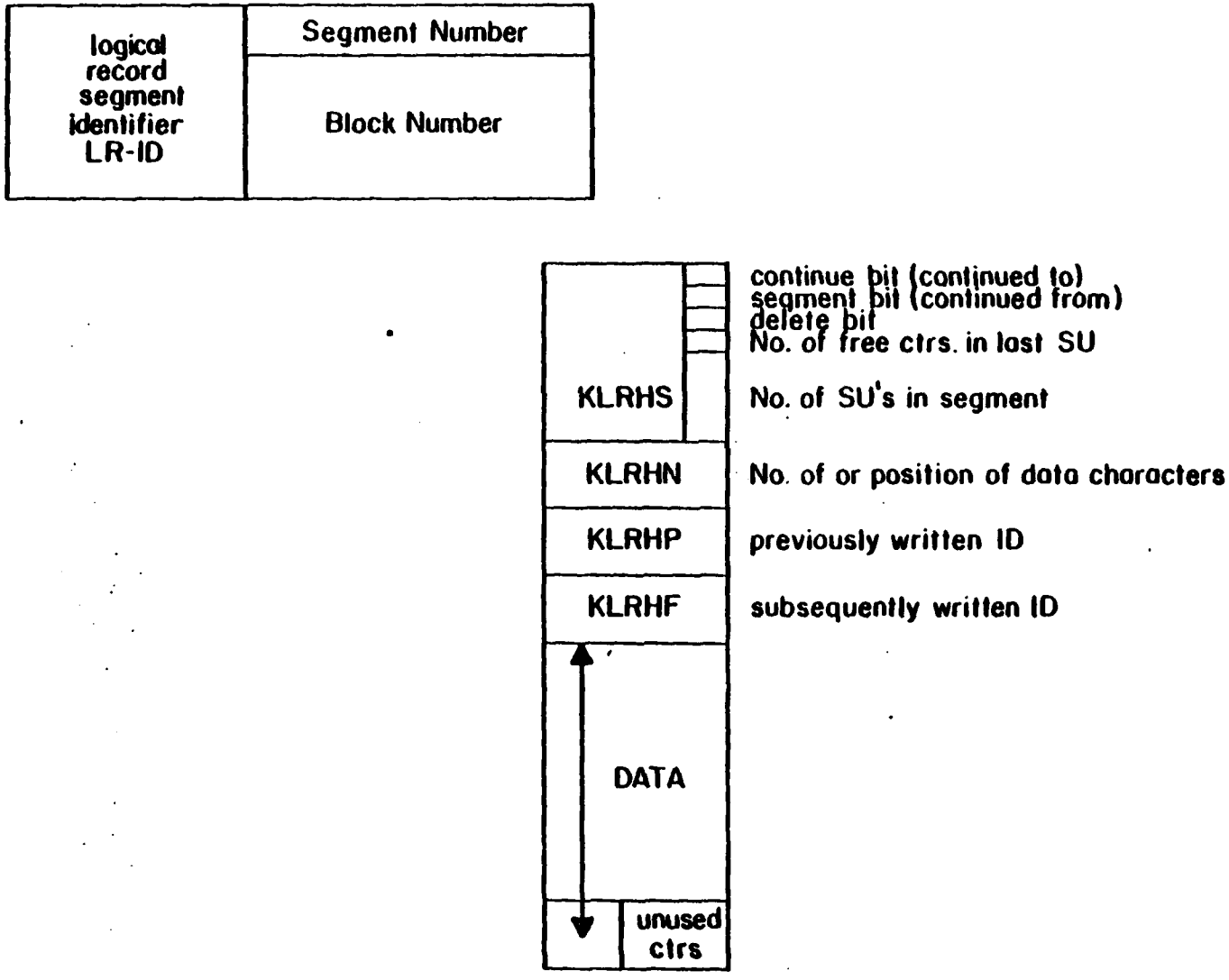


Figure 6. Logical Record Segment Layout

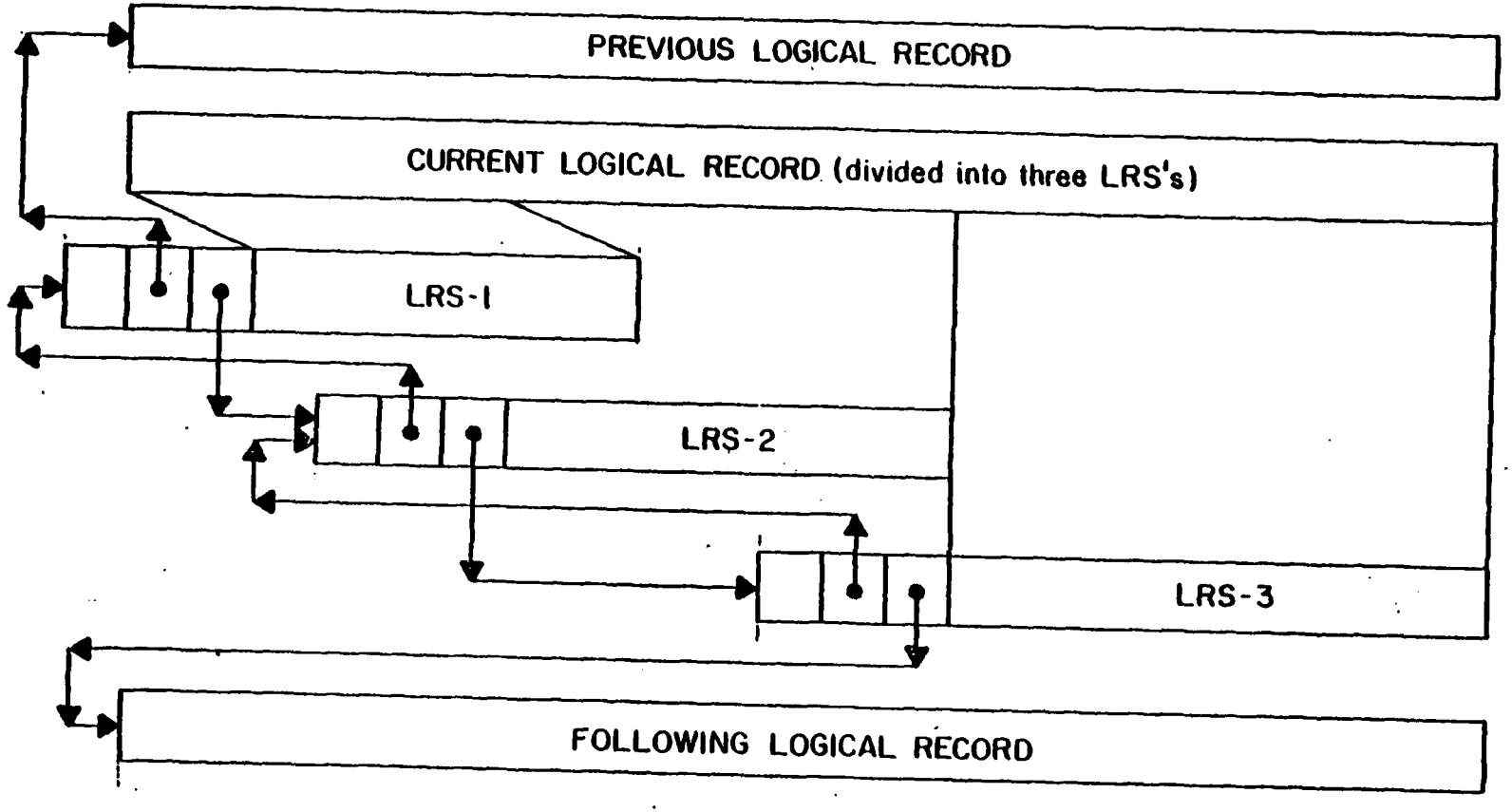


Figure 7. Logical Record Segment Chain

TABLE 1

FIXED AREA

Variable/ Array	Position in COMMON	Description	Value
		1. SYSTEM WIDE	
	1		
	2		
	3		
	4		
		*** Machine Characteristics ***	
NBW	5	Number of bits per word	
NCW	6	Number characters in a word	
NAW	7	Number of addresses in a word	
NBC	8	Number of bits per character	
NCU	9	Number of characters in a "standard unit" (approximately 32 bits long)	
NBU	10	Number of bits per standard unit	
NAU	11	Number of addresses in a standard unit	
NBU2	12	One-half of NBU	
NC2U	13	Number of characters in a double precision variable (about 8-10 chars)	
RVRSE	14	Flag telling order of characters: 0 = left to right 1 = right to left	
NWPSU	15	Number of real words per standard unit: 0 = two SU's per real (e.g. IBM 1130) 1 = one SU per real (e.g. IBM 360) 2 = two real's per SU	

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
LFIW	16	Number of bits in a FORTRAN integer	
LFIWU	17	Number of useable bits in a FORTRAN integer	
	18		
	19		
	20		
	21		
	22		
	23		
		*** Standard Devices ***	
ICRD	24	Card reader logical unit number	
IPRT	25	Line printer logical unit number	
IPCH	26	Card punch logical unit number	
ITAP	27	Magnetic tape logical unit number	
ITRMI	28	Terminal input logical unit number	
ITRMO	29	Terminal output logical unit number	
JTRMI	30	Second terminal input logical unit number	
JTRMO	31	Second terminal output logical unit number	
IPLT	32	Plotter logical unit number	
IGRAI	33	Interactive graphics input logical unit	
IGRAO	34	Interactive graphics output logical unit	

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
	35		
	36		
	37		
		*** Constants and Masks ***	
TRU	38	Value of logcial true (largest positive integer)	
FALS	39	Value of logical false (largest negative integer)	
ZEROS	40	A word of all "zeros"	
ONES	41	A word of all "ones"	
FORM1	42	Token format mask (1,2,1)	
FORM2	43	CDL dictionary format mask (2,2)	
FORM3	44	CDB pointer format mask (3,1)	
	45		
	46		
	47		
	48		
	49		
		*** Executive Data Area ***	
SCA	50	Subscript of start of communications area (=176)	
ATLID	51	LRID of active task list	
SCRFL	52	Name of executive work file	
SCRFL	53	Name of executive work file	

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
	54		
		*** Lexical Analyzer Data Area ***	
NXCHR	55	Next character in input buffer	
IDFT	56	Data field type	
ISTD	57	Start of data field	
LSTR	58	Length of string or an integer value	
DCOM	59	Command (up to 8 characters) or a	
DCOM	60	double precision value	
MLCA	61	Maximum length of user communications area (for referencing in a CDB)	
SYDID	62	LRID of system command dictionary	
DICID	63	LRID of current command dictionary	
CDBNM	64	Name of current CDB	
CDBNM	65	Name of current CDB	
CDBID	66	LRID of current CDB	
SYMID	67	LRID of current symbol table	
SVTID	68	LRID of current standard value table	
NXELM	69	Pointer to next CDB element	
RELOC	70	Subsystem common relocation factor (must be a positive number)	
CSTID	71	LRID of modifier condition stack	
WATID	72	LRID of ID wait list	
	73		
	74		

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
	75		
	2.	DATA HANDLER INTERNAL VARIABLE AND ARRAYS.	
IPOOL	76	Position in COMMON of beginning of pool area	201
IPLEN	77	Length of pool area	9800
NBLMX	78	Maximum length of a block	256
NBSGN	79	Number of bits in a segment number	8
NBBLK	80	Number of bits in a block number	24
ILLRH	81	Length of an LR (segment) header	4
ILRHS	82	Position in LR (segment) header of KLRHS	1
ILRHN	83	Position in LR (segment) header of KLRHN	2
ILRHP	84	Position in LR (segment) header of KLRHP	3
ILRHF	85	Position in LR (segment header of KLRHF	4
ILPRH	86	Length of a PR header	3
IPRHS	87	Position in PR header of KPRHS	1
IPRHP	88	Position in PR header of KPRHP	2
IPRHF	89	Position in PR header of KPRHF	3
IBTST	90	0'th displacement of BT	(computed)

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
NEBT	91	Number of BT entries	(computed)
IELBT	92	Length of a BT entry	273
ILBTH	93	Length of a BT header	17
IBTNS	94	Position in BT header of KBTNS	1
IBTBL	95	Position in BT header of KBTBL	2
IBTQ	96	Position in BT header of KBTQ	3
IBTPP	97	Position in BT header of KBTPP	4
IBTL	98	Position in BT header of KBTL	5
IBTTR	99	Position in BT header of KBTTR	6
IBTCN	100	Position in BT header of KBTCN	7
IBTDP	101	Position in BT header of KBTDP	8
IBTNC	102	Position in BT header of KBTNC	9
IBTSW	103	Position in BT header of KBTSW	10
IBTFN	104	Position in BT header of KBTFN	11
IBTFI	105	Position in BT header of KBTFI	15
IBTPI	106	Position in BT header of KBTPI	16

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
IBTLN	107	Position in BT header of KBTLN	17
IBTLG	108	0'th displacement of BT entry referenced as result of most recent call of DKGET	(variable*)
IBTPQ	109	0'th displacement of BT priority queue	(computed)
IBTOP	110	Position in BT of entry with highest priority	(variable**)
IBTBT	111	Position in BT of entry with lowest priority)	(variable***)
IOFST	112	0'th displacement of OFT	(computed)
NEOFT	113	Number of OFT entries	125
IELOF	114	Length of an OFT entry	9
IOFFN	115	Position in OFT entry of KOFFN	5
IOFFL	116	Position in OFT entry of KOFFL	6
IOFLL	117	Position of OFT entry of KOFNB	7
IOFNB	118	Position in OFT entry of KOFNB	8
IOFLB	119	Position in OFT entry of KOFLB	8
IOFFD	120	Position in OFT entry of KOFFD	9
IDSST	121	0'th displacement of DST	200
IELDS	122	Length of a DST entry	12

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
IDSLU	123	Position in DST entry of KDSLU	1
IDSFN	124	Position in DST entry of KDSFN	2
IDSPR	125	Position in DST entry of KDSPR	4
IDSPB	126	Position in DST entry of KDSPB	5
IDSPL	127	Position in DST entry of DKSPL	6
IDSFD	128	Position in DST entry of KDSFD	8
IDSFS	129	Position in DST entry of KDSFS	9
IDSSU	130	Position in DST entry of KDSSU	7
ILDSN	131	Length of a data set name	2
OPSYS(2)	132	Name of host operating system	'IBM OS'
DHVER	134	DH version	'V1F '
DHFLG	135	DH flag	'DH '
NBCON	136	Position in KLRHS of continue bit	29
NBSEG	137	Position in KLRHS of segment bit	30
NBDEL	138	Position in KLRHS of delete bit	31
NBWIN	139	Position in KBTSW of written-into bit	0

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
NBRIT	140	Position in KBTSW of write-in-progress bit	1
NBRED	141	Position in KBTSW of read-in-progress bit	2
NBOPN	142	Position in KDSLU of open bit	16
NBOLD	143	Position in KDSLU of old bit	17
NBSAV	144	Position in KDSLU of save bit	18
NBSHR	145	Position in KDLSU of share bit	19
NBCHN	146	Position in KOFFD of chain bit	0
NBPAK	147	Position in KOFFD of pack bit	1
NBPRO	148	Position in KOFFD of protect bit	4
NBNDS	149	Bit position in KOFFD of beginning of data set number	8
K0	150	Zero	0
K1	151	One	1
K2	152	Two	2
IDSBI	153	Position in DST entry of KDSBI	10
IDSWN	154	Position in DST entry of KDSWN	11
IDSWD	155	Position in DST entry of KDSWD	12

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
HEDLR(4)	156	Work area for an LR (segment) header	(variable)
DST(12)	156	Work area for a DST entry	(variable)
DATA(17)	160	Work area for a BT header	(variable)
		3. DATA HANDLER COMMUNICATIONS.	
STASK	176	Sending task name (RSX-11D scratch area)	
LCA	177	Length of user communications area transmitted by last/next message set	
TRACE	178	Executive trace switch: each bit, when on, indicates a type of tracing to be printed on unit IPRT.	

Bit	(Value)	Item Printed
0	1	Digitize expansions
1	2	Tokens created
2	4	Macro expansions
3	8	Macro argument Substitions
4	16	Names of programs/tasks executed
5	32	Names of CDB's called
6	64	Statements of CDB's interpreted
7	128	Common values modified
8	256	Macro processor return information
9	512	Command dictionary search information
10	1024	
11	2048	
12	4096	
13	8192	
14	16384	
15	32768	

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
EXST1	179	Executive status switch 1: Each bit, when on. signifies the presence of some exceptional condition.	
		Bit (Value) Meaning	
		0 1 Return to executive	
		1 2 ID labels required	
		2 4 ID wait list processing	
		3 8 Inhibit user program execution (system control)	
		4 16 Inhibit user program execution (user control)	
		5 32 Next record continuation of preceeding command	
		6 64 Repeat tabular processing	
		7 128 End of file on input	
		8 256 Input error occurred (CI)	
		9 512	
		10 1024 User program signaled error	
		11 2048	
		12 4096 Interactive mode	
		13 8192	
		14 16384	
		15 32768	
	180		
PSTID	181	LRID of program/CDB stack	
TORID	182	LRID of token list	
TVLID	183	LRID of token value table	
NXTKN	184	Next token to be processed	
	185		
INCMD	186	Logical unit number from which to read commands	
		*** Data Handler Arguments ***	
MAXGP	187	Maximum number of characters per GET/PUT	

TABLE 1 (CONT'D)

Variable/ Array	Position in COMMON	Description	Value
NDS	188	Data set number	
FILNM	189	File name (up to 8 characters)	
FILNM	190	File name (up to 8 characters)	
LULID	191	LRID of next (or first) record in file	
PREID	192	LRID of preceeding (or last) record	
LRLEN	193	Logical record length in characters	
FSTAT	194	File status (from DKFIL)	
NDA	195	Maximum number of direct access data set	
IDHTR	196	Data handler trace switch	
IDHER	197	User error handling switch	
IDKER	198	Data handler error number	
SSYSN	199	Subsystem name (up to 8 characters)	
SSYSN	200	Subsystem name (up to 8 characters)	

* initialized to value of IBTST
** initialized to 1
*** initialized to value of NEBT

(2) If this LRS is not the first LRS, LRS ID of the LRS written previously for this LR.

Chain Option

(1) If this LRS is the first one in the file, zero.

(2) If this is the first LRS of a LR, LR ID for the first LRS of the previously written.

KLRHF contains the following:

No Chain Option

(1) If LRS not the last LRS in LR, LRS ID of the subsequent LRS.

(2) If LRS is the last LRS in LR, or the point option is not in effect, zero.

Chain Option

(1) If LRS is the last LRS in CR, LR ID for the first LRS of the subsequent record.

The actual data is stored next followed by the unused characters in the last SU.

Program Common Structure

The great majority of internal variables and arrays used by the DH are located in blank CØMMØN (Figure 8). Common consists of a fixed area and a pool (dynamic) area. The first sections in the fixed area contain system-wide and DH communications variables and arrays. The second section contains the DH internal variables and arrays. All three sections are documented in Table 1. All positions and lengths in Table 1 are given in terms of SU's. All elements in this section that need to be initialized are done so during DH initialization.

The pool area contains the four DH tables: the data set table (DST), the open file table (ØFT), the buffer table (BT), and the BT priority queue. There is an unused area at the end of common.

The DST contains an entry for each data set that is currently in use by the DH (Figure 9). The high-order half of KDSLU contains four status bits known as the open, old, save, and share bits, whose positions are given by the CØMMØN variables NBØPN, NBØLD, NBSA, and NBSHR, respectively (Table 1). The open bit indicates whether or not the data set is open to the host operating system. The old bit indicates whether or not the data set existed before the current

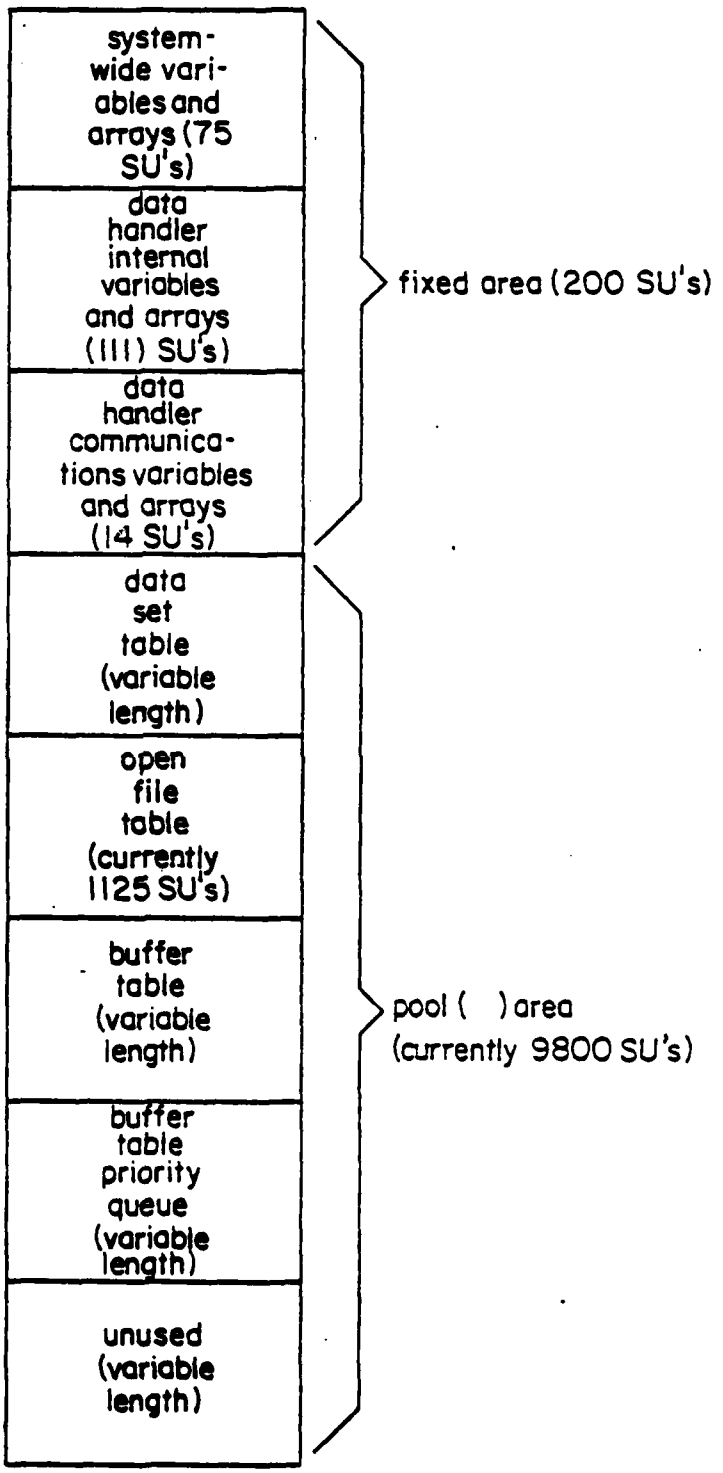


Figure 8. Common Layout

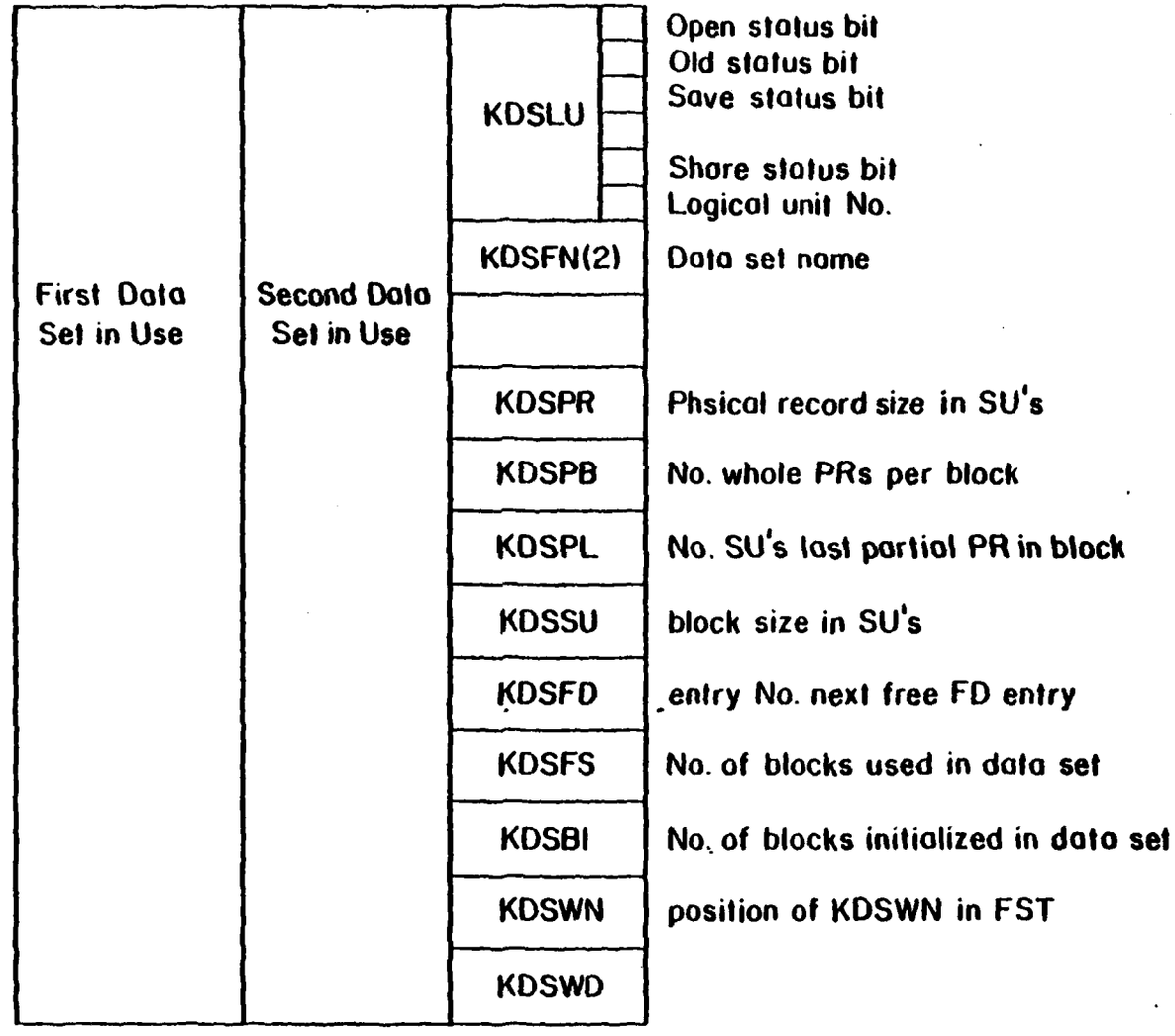


Figure 9. Data Set Table Layout

execution of the DH. The save bit indicates whether or not the data set is to be saved at the end of the current execution. The share bit indicates whether the data set can be shared with another, concurrently-executing DH task. The low-order half of KDSLU is the logical unit number.

KDSFN is the data set name, KDSPR is the PR size (in SU's), KDSPB is the number of whole PR's per block, KDSPL is the number of SU's in the last (partial) PR of a block if the block does not end on a PR boundary (it is zero otherwise), KDSSU is the block size (in SU's), KDSFD is the entry number of the next free FD entry, KDSFS is the number of blocks initialized in the data set, and KDSWN is the postion of KDSWN within the FST.

The ØFT contains an entry for each file that is currently in use for output by the DH (Figure 10). KØFFN(4) is the file name. KØFFL and KØFLL are the LR ID's of the first and last LR's, respectively, that were written for the file. KØFNB is the block number of the next block to be added to the file. KØFLB is the block number of the last block that was written for the file. The high-order half of KØFFD is the entry number of the FD entry associated with this file. The bits in the low-order half of KØFFD constitute the data set number (DST entry number) of the data set to which the file belongs. The extreme low-order portion of the low-order half of KØFFD contains three status bits known as the chain, pack, and protect bits, whose positions are given by the CØMMØN variables NBCHN, NBPAK, and NBPRØ. The chain and pack bits indicate whether or not the chain and pack options are in effect for the file. The protect bit indicates whether or not write protection is in effect for the file.

The BT contains an entry for each block of a data set that is currently in main storage (Figure 11). KBTNS is the data set number (DST entry number) of the data set to which this block belongs. KBTBL is the block number of the block. KBTQ is the segment number of the segment which was requested when the block was read into main storage, and KBTPP is the position in CØMMØN of the SU immediately proceeding the beginning of the segment. KBTL and KBTTR are the segment and block numbers, respectively, of the LR that was most recently referenced in this BT entry. KBTCN is the position within this LR of the first (data) character (of a set) that was requested. KBTDP is the position in CØMMØN of the SU that contains this character. KBTNC is the number of characters in the first segment of this LR. KBTSW contains three status bits known as the written-into, write-in-progress, and read-in-progress bits, whose positions are given by the CØMMØN variables NBWIN, NBWIN, NBRIT, and NBRED, respectively. The written-into bit indicates whether or not the block has been modified in this BT entry. The write-in progress bit indicates whether or not an output operation is in progress for the block. The read-in-progreas bit indicates whether or not an input operation is in progress for the block. KBTFN(4) is the name of the file to which the block belongs. KBTFI and KBTPI have the same values as KLRHF and

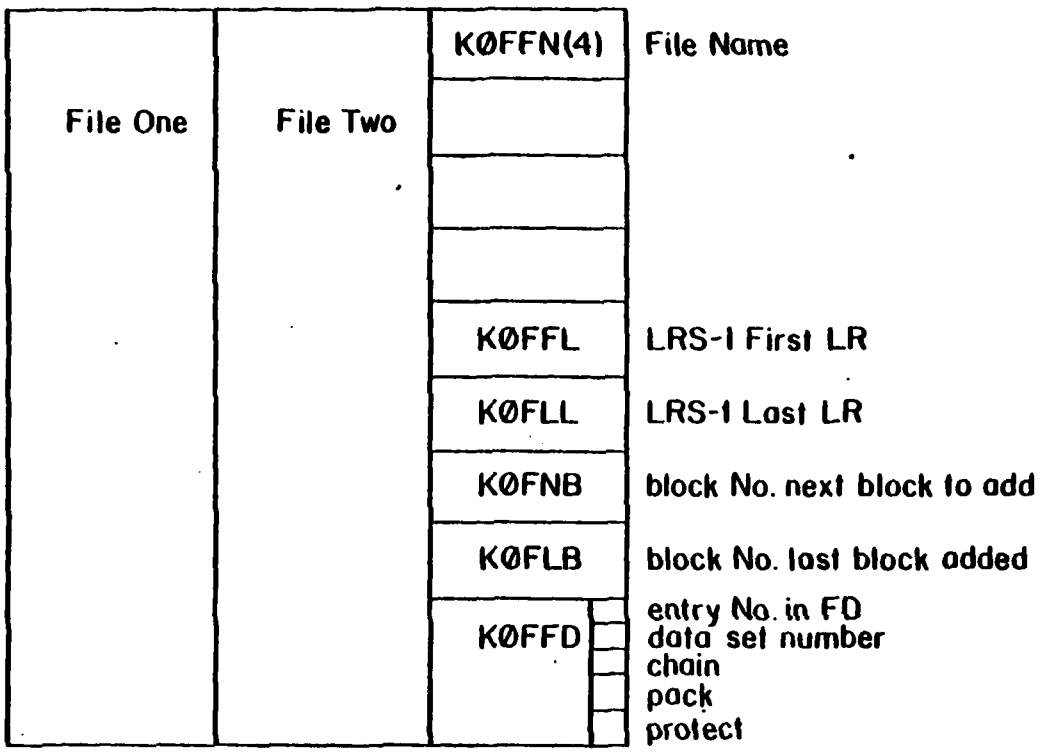


Figure 10. Open File Table/File Directory Entry Layout

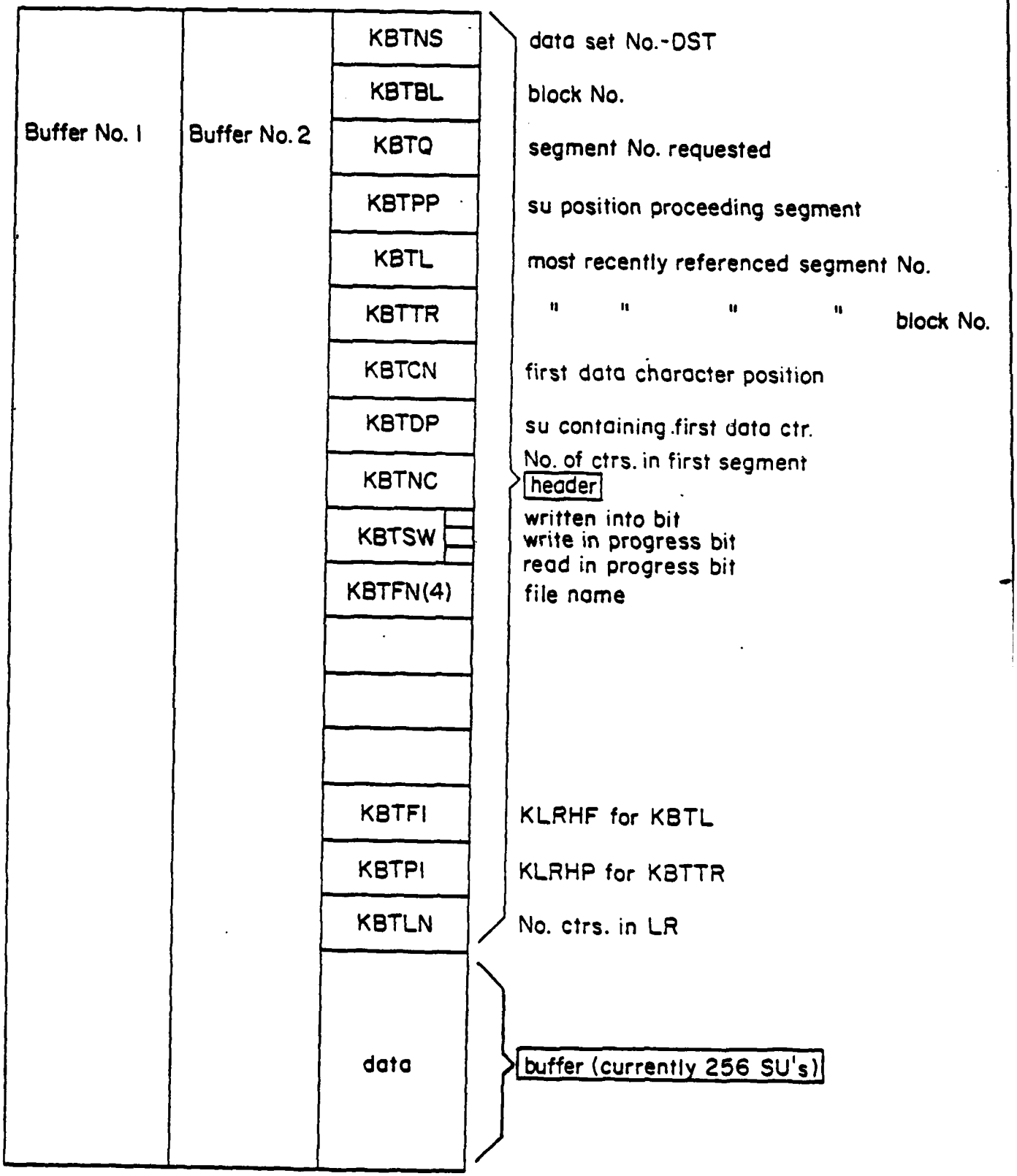


Figure 11. Buffer Table Layout

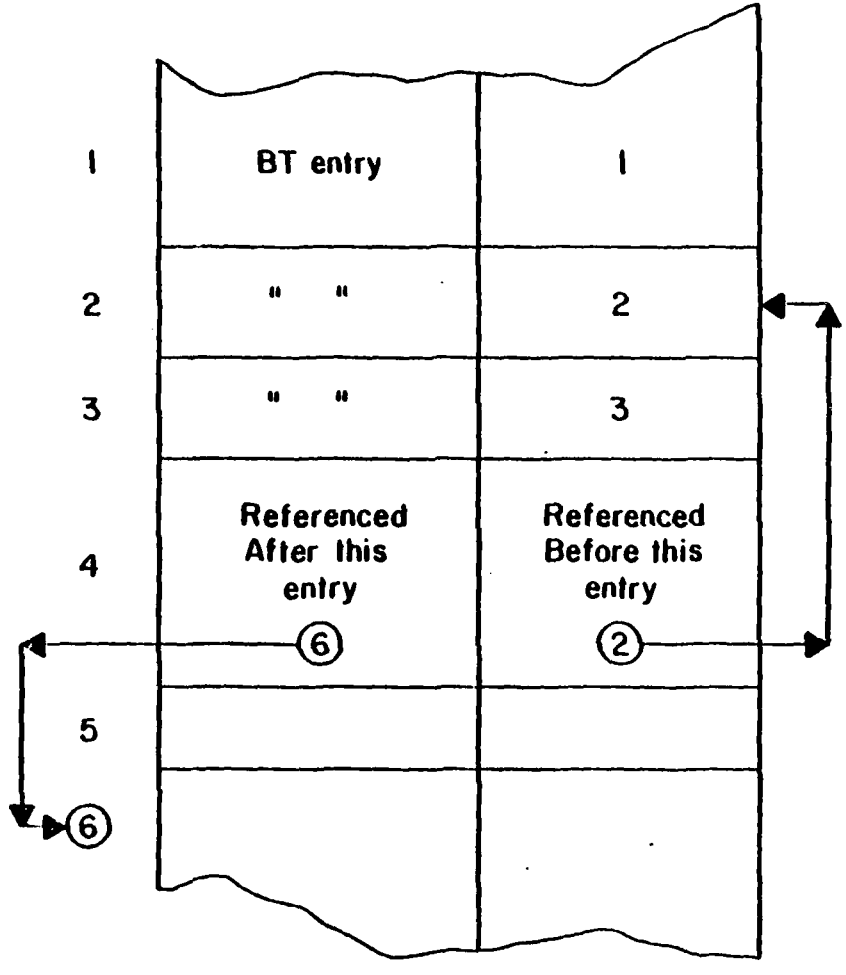


Figure 12. BT Priority Que Layout

KLRHP, respectively, in the LR header of the LR given by KBTL and KBTTR. KBTLN is the number of characters in the LR.

The BT priority queue contains an SU entry for each BT entry (Figure 12). The high-order half of the i'th BT priority queue entry is the entry number of the BT entry that was referenced immediately after the i'th BT entry. The low-order half of the i'th BT priority queue entry is the entry number of the BT entry that was referenced immediately before the i'th BT entry. Hence, the BT priority queue imposes a most-recently-referenced-to-least-recently-referenced ordering on the BT entries. A more recently referenced BT entry is said to have a higher priority than a less recently referenced BT entry. A block in a higher-priority BT entry will be retained in main storage longer than a block in a lower-priority one.

The DH includes a routine which produces a dump of blank CØMMØN. All items are identified in the same manner as they are in this documentaion. The values are given in decimal digits and characters as applicable except for KDSLU, KØFFD, KBTSW, the buffer contents of the BT entries, and the BT priority queue entries, which are given in hexadecimal digits. An annotated dump is given in Appendix A.

Definitions

Data Set Block - Fixed-sized unit of information stored on a data set. The actual information read (or written) from the disk and to the local program I/Ø buffer area.

Program I/Ø Buffer - Fixed-sized unit of storage in the local program space to be used to store data read from (to be written to) disk storage.

Physical Record - Fixed-sized unit of information measurement which must be equal to or less than the size of the data set block.

2 APPLYING THE DATA-HANDLER

The use of the data handler (DH) within a computer system application program requires the review and application of two data handler routines. The first routine, DKNIT, initializes the data handler, allows the system programmer to define the system parameters to the data handler, and builds the necessary tables for DH execution.

The system programmer must review the definition of each parameter and variable in this routine and define the values with respect to the actual operating system characteristics.

This routine must be the first routine called by the application system.

When all processing has been completed the application system should call DKXIT to logically close all DH managed data sets.

During the period after the call to DKNIT and before DKXIT all application programs can utilize the DH application routines to read and/or work information from/to the DH datasets.

3 APPLICATION COMMAND SUBROUTINE DESCRIPTION

Definitions

Several parameters are kept in COMMON, rather than being passed as arguments. These include:

FILENAME - the name of the file being operated on; it contains up to 16 characters.

NDS - the data set number, usually 1,2,3, or 4.

PREID - the logical record identifier of the preceding record in the file.

FOLID - the logical record identifier of the following record in the file.

LRLEN - the logical record length, in characters.

FSTAT - the file status (see DKFIL 4.8).

Learning the Commands

The logical order for learning the commands and functions is to read the descriptions in the following order:

1. Access to Data.

1. LOK/UNLOK - to obtain controlled access to information.

2. Data Sets.

1. DKNNDS - To find a data set number.
2. DKDSN - To find data set name
3. DKSET - To obtain general information.

3. Files.

1. DKOPN - To create and/or open a file for writing.
2. DKPUT - To write information.
3. DKINS - To write (insert) information
4. DKGET - To read information
5. DKDEL - To delete information.
6. DKRNM - To rename the file
7. DKFIL - To obtain general information
8. DKLEN - To obtain the length of a record.
9. DKLOS - To disallow writing

PURPOSE

To delete a specific record within a file or to delete a complete file.

GENERAL FORM

```
CALL DKDEL (RECID)
```

where RECID is a zero or the record id for an existing record. (INPUT)

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
1. RECID	a. valid record id b. zero (0) to delete all records in the file FILENAME	No Default

SPECIAL NOTES

The programmer must set variables NDS and FILENAME in common before the subroutine is called.

EXECUTION PROCEDURE

The system will delete the record specified from the data set NDS. If the RECID is zero, the entire file defined in FILENAME will be deleted from data set NDS.

PURPOSE

To find the dataset name when the dataset number is known.

GENERAL FORM

CALL DKDSN(DSNM)

where | is returned as the dataset name (output)

FIELD OPTIONS - None

SPECIFIC NOTES

The programmer must set variables NDS and FILENAME in common before the subroutine is called.

EXECUTION PROCEDURES

The system will locate the data set specified by NDS and return the dataset name. If the data set is not found, the routine will return the next data set with a larger NDS valve. NDS will be changed to point to the new data set. If no more data sets are available, NDS and DSNM will be zeroed.

OBTAIN FILE INFORMATION
DKFIL

PURPOSE

To obtain the general information about a specified file.

GENERAL FORM

```
CALL DKFIL
```

FIELD OPTIONS - None

SPECIAL NOTES

The programmer must define NDS and FILENAME in common before this subroutine is called.

EXECUTION PROCEDURES

FSTAT is returned in common as
1 - if the file does not exist
2 - if the file is closed
3 - if the file is open

PREID is returned in common as the record id of the first record in the file.

FOCID is returned in common as the record id of the last record in the file.

PURPOSE

To obtain information from a file on disk storage.

GENERAL FORM

```
CALL DKGET (RECID, AREA, L1, L2, NC)
```

where	RECID	is the logical record identifier. (INPUT)
	AREA	is the location into which the information is to be placed. (OUTPUT)
	L1	is the first character to be read from the record. (INPUT)
	L2	is the last character to be read from the record. (INPUT)
	NC	is the actual number of characters read from the record. (OUTPUT)

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
1. RECID	a. valid record id	No Default

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
2. AREA	a. one computer word, variable name. b. the first computer word to be used to hold the data. An element of a dimensioned array.	No Default
3. L1	a. less than one or greater than the length of the logical record, it is interpreted as being equal to one. b. the first character to be read.	No Default
4. L2	a. greater than the number of characters in the specified logical record or if less than L1, it is interpreted as being equal to the last character in the record. b. the last character to be read. c. zero - is interpreted as being equal to the last character in the record.	No Default
5. NC	Actual number of of characters read.	No Default

SPECIAL NOTES

The programmer must set NDS in common before a call to this subroutine. The programmer can read an entire logical record of unknown length by setting L1 and L2 equal to zero. The programmer must define RECID, AREA, L1 and L2.

EXECUTION PROCEDURES

The system will read the requested information (L1, L2) from the requested record RECID located on data set NDS. The data will be stored starting with word AREA. The system will return NC as the number of characters actually placed into AREA.

PURPOSE

To insert a record at the beginning of a point file or to insert a record after a specified record in a point file.

GENERAL FORM

```
CALL DKINS (RECID, AREA, L1, L2)
```

where	RECID	is the logical record identifier after which the new record is to be entered. (INPUT) Is the record id of the new record; (OUTPUT)
	AREA	is the first word of the information to be placed into the new record. (INPUT)
	L1	is the first character in the record to receive the first character in AREA. (INPUT)
	L2	is the last character in the record to receive information from AREA. (INPUT)

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
1. RECID	a. record id after	No Default

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
	which the new record will be placed. b. zero to indicate placement as the first record in the file. c. computer output. record id of new data record.	
2. AREA	the location in core from which the writing will start.	No Default
3. L1	a. is the first character in the record to receive the first character in AREA. b. zero to indicate start with the first chracter.	No Default
4. L2	is the last character in the record to receive information from AREA.	No Default

SPECIAL NOTES

The programmer must set NDS in common. If the programmer is writing a new record, FILENAME must be set in common. The programmer must define RECID, AREA, L1 and L2 before calling this subroutine.

EXECUTION PROCEDURES

A new record will be created after record RECID. If RECID is zero, a new record will be added to the beginning of the file. The system will place the first character in AREA in the L1 character location in the new record. Characters will be copied from AREA until the L2 character has been written into the new file. The record id for the new record will be returned in RECID.

OBTAIN LOGICAL RECORD LENGTH
DKLEN

PURPOSE

To obtain the logical record length of an existing record.

GENERAL FORM

```
CALL DKLEN (RECID)
```

where RECID is an existing logical record id. (INPUT)

FIELD OPTIONS - None

SPECIAL NOTES

The programmer must set NDS in COMMON before calling this routine.

EXECUTION PROCEDURES

The system will obtain the length of record RECID in data set NDS and return it in variable LRLEN in common. The system sits PREID and FOLID also.

CLOSE A FILE
DKLOS

PURPOSE

To make a file unavailable for writing.

GENERAL FORM

```
CALL DKLOS
```

FIELD OPTIONS - None

SPECIAL NOTES

The programmer must set NDS and FILENAME in common before calling this routine.

EXECUTION PROCEDURES

All records in core for FILENAME are marked for writing and will be placed onto the disk NDS. The file will be marked as closed and no one will be allowed to write into the file until another DHOPN is issued.

PURPOSE

To find the data set number when the data set name is known.

GENERAL FORM

```
CALL (DSNAM, DSNO)
```

where	DSNAM	is the name of the data set. (INPUT)
	DSNO	is the number of the data set. (OUTPUT)

FIELD OPTIONS - None

SPECIAL NOTES - None

EXECUTION PROCEDURES

The system searches the data set name list and returns the data set number DSNO. If the dataset is not found, DSNO will be returned as zero.

DATA HANDLER INITIALIZATION
DKNIT

PURPOSE

To initialize the data area in core for the data-handler and to establish the dataset name - logical reference number association.

GENERAL FORM

```
CALL DKNIT
```

FIELD OPTIONS - None

SPECIAL NOTES

This routine should be called once at the beginning of the application system.

EXECUTION PROCEDURES

The system will establish all in core data elements prior to the first use of any other data-handler subroutine.

PURPOSE

To open a new or existing file.

GENERAL FORM

```
CALL DKOPN (IOP)
```

where IOP is the storage option code for a new file only. (INPUT)

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
1. IOP	a. new files only 0 - neither point nor pack 1 - point only 2 - pack only 3 - point and pack b. not used for existing files	No Default

SPECIAL NOTES

The programmer must set NDS and FILENAME in COMMON. IOP is only applied if the file is new.

EXECUTION PROCEDURES

The file FILENAME is opened for writing in data set NDS using the original storage option given by the original DKOPN. If the file does not exist, the file is created.

WRITE INFORMATION TO A RECORD
DKPUT

PURPOSE

To write information to a record.

GENERAL FORM

```
CALL DKPUT (RECID, AREA, L1, L2)
```

where	RECID	is a record id in a data set. (INPUT) (OUTPUT)
	AREA	is the information to be written. (INPUT)
	L1	is the first character in the record to receive the first character in AREA. (INPUT)
	L2	is the last character in the record to receive data from AREA. (INPUT)

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
1. RECID	a. zero - to write a new record at the end of a file. (INPUT) b. a valid record id of an existing record. (INPUT) c. record id of the record written into.	No Default

FIELD OPTIONS

FIELD	OPTIONS	DEFAULT
	(OUTPUT)	
2. AREA	a. one computer word, variable name b. the first computer word to be used to hold the data. An element of a dimensioned array.	No Default
3. L1	first character in the record to receive information.	No Default
4. L2	a. positive - last character in the record to receive information from AREA. b. negative - truncation of the logical record will occur. The last character will be the absolute valve of L2.	No Default

SPECIAL NOTES

The programmer must set NDS and FILENAME in COMMON.

EXECUTION PROCEDURE

If RECID is initially zero, a new record at the end of the file will be generated. The first character in AREA will be placed in the L1 character position in the record. Copying will continue until a character has

been written into the absolute value of L2. If L2 is negative, the existing record will be reduced to L2 characters.

PURPOSE

To change the name of an existing file.

GENERAL FORM

```
CALL DKRNM (OCDNM, NEWNM)
```

where	OLDNM	is the current file name. (INPUT)
	NEWNM	is the new file name. (INPUT)

FIELD OPTIONS - None

SPECIAL NOTES

The programmer must set NDS in COMMON before applying this routine.

EXECUTION PROCEDURE

The old name will be changed to the new name.

PURPOSE

To obtain the current information related to a specific document.

GENERAL FORM

```
CALL DKSET (LBLK, NBU, NTALL)
```

where	LBLK	is the block length in standard units. (OUTPUT)
	NBU	is the number of blocks used. (OUTPUT)
	NTALL	is the total number of blocks used. (OUTPUT)

FIELD OPTIONS - None

SPECIAL NOTES

The programmer must set NDS before calling this routine.

EXECUTION PROCEDURE

The system will set all three parameters and return.

END DATA HANDLER
DKXIT

PURPOSE

To clear all batters and write out all changes to the disks.

GENERAL FORM

```
CALL DKXIT
```

FIELD OPTIONS - None

SPECIAL NOTES

This routine should be called once before exiting from the application system.

EXECUTION PROCEDURES

The system will logically end all processing of data. All batters will be cleared and data change written to disk.

PURPOSE

To obtain access to existing files.

GENERAL FORM

```
CALL LOCK (IND)
CALL UNLOK
```

where IND indicates control requested

FIELD OPTIONS

FIELD	OPTION	DEFAULT
1. IND	a. 0 - unconditional b - 1 - conditional	No Default

SPECIAL NOTES

During the actual execution of the data-handler in an interactive or multiprogramming environment, several users may be executing different copies of the code at the same time. Some of the users may wish to access (to read and/or write) the same files at the same time. The "multi-user" feature will permit concurrently executing programs to share the use of files. Shared use of these resources must be strictly controlled in order to ensure that one program does not interfere with the correct execution of others. This control takes the form of synchronizing use of these resources on the part of the programs invovled.

When a program requires use of a resource, it must request control of that resource from the operating system. Control of a resource can be either exclusive or shared. Exclusive control of a resource guarantees that no other program will be granted access to that resource (write access). Shared control guarantees that no other program will be granted exclusive control of that resource, but other programs will be granted shared control of that resource (read only access).

A request for either kind of control of a resource can be either conditional or unconditional. For conditional request, control is granted only if the resource is immediately available. The requesting program is informed as to whether or not control was granted. For unconditional requests, control is granted as soon as the resource becomes available. The requesting program may have to wait for an indefinite amount of time. This eventuality is entirely transparent to the program itself.

When a program has finished using a resource, it must relinquish control of that resource so that it becomes available for use by other programs.

REQUESTING AND RELINQUISHING CONTROL OF RESOURCES IS THE PROGRAMMER'S RESPONSIBILITY. Two subroutines, LOCK and UNLOK, have been made available for this purpose.

Required Programming Before Calling LOCK or UNLOK

```
INTEGER RSRCS
COMMON /LOCKC/ RSRCS (7,40), NRSRCS
```

Before calling LOCK or UNLOK, you must describe the resources to be requested or relinquished via the array RSRCS. Each column corresponds to one resource. The number of resources described in RSRCS must be defined in the variable NRSRCS.

Before a call to LOCK, RSRCS(2,I), RSRCS(3,I), RSRCS(4,I), and RSRCS(5,I) must contain the name of the I'th resource requested. RSRCS(1,I) must contain the number of the data set on which this resource resides. RSRCS(7,I) should = 1 if exclusive control of this resource is requested, 0 if shared control is requested. NRSRCS should contain the number of resources, RSRCS(6,I) should not be used within a processing program.

LOCK, as its name implies, is to be used to request control of resources.

LOCK has a single argument, the integer variable IND, which should = 1 if the request is conditional, 0 if the request is unconditional. If IND = 1, LOCK will return the status of the request via the argument IND. IND = 1 if control of all requested resources was granted, 0 if control over none of the resources was granted due to the non-availability of one or more of the resources. If the request was unconditional, then control of all requested resources was granted (IND was not modified and still = 0). Note that LOCK (or UNLOK) never modifies LOCKC.

UNLOK is to be used to relinquish control of resources. UNLOK has no arguments, since control of resources is always relinquished unconditionally.

Application Rules

The following rules must be strictly observed:

(1) Control of a resource should not be requested until it is needed;

(2) A resource must not be used until after its control has been requested and granted;

(3) a. The use of DKOPN, DKPUT, DKINS, DKLOS, DKSET, DKRNM, DKXIT, or DKCLR in connection with a resource requires EXCLUSIVE CONTROL of that resource;

b. The use of DKFIL, DKLEN, or DKGET in connection with a resource requires shared or exclusive control of that resource;

c. The use of DKNDS, DKNIT, or DKDMP is unrestricted;

(4) Control of a resource must not be relinquished until it is consistent with respect to both itself and other resources;

(5) Control of a resource should be relinquished as soon as it is no longer needed.

(6) A resource must not be used after its control has been relinquished;

(7) The logic of execution should be such that: UNLOK is not

called without a previous call to LOCK; two calls to LOCK/UNLOK are never made without an intervening call to UNLOK/LOCK; LOCK is not called without a subsequent call to UNLOK.

(8) a. The same resource must not be referenced by two different columns in RSRCS;

b. The maximum number of resources to be locked is 40;

(9) No assumptions may be made regarding the contents of a resource at the time its control is granted.

Failure to observe one or more of the above rules will not necessarily result in the occurrence of a perceptible error condition (such as an ABEND), but resource integrity may nevertheless be seriously impaired. It is most important that great care in coding be exercised.

4 SUBROUTINE FUNCTIONS

Appendix B briefly describes the functions performed by each subprogram. This list is ordered by subroutine name.

Subroutines are divided into two general types: (1) application subroutines, and (2) function subroutines. The user's apply application programs application subroutines to request the data handler to perform work. The application and function subroutines use the function subroutines to actually perform work within the data handler.

5 ERROR MESSAGES

The following data-handler errors are defined below:

DATA-HANDLER PACKAGE ERROR NUMBERS PAGE 1

ERROR NO.	ROUTINE	CONDITION
1	DKNIT	- SYST COMMON NOT INITIALIZED
2	DKNIT	- NOT ENOUGH SPACE FOR 1 BUFFER
3	DKGO	- NBSGN + NABLK .NE. UBU
4	DKODS	- OLD FILE NOT A DH FILE
5	DKODS	- CAN'T PROCESS OLD FILE - BLOCK SIZE TOO BIG
6	DKODS	- ERROR READING OLD FILE HEADER
7	DKODS	- ERROR OPENING OLD FILE
8	DKODS	- ERROR OPENING NEW FILE
9	DKARG	- NDS INVALID
10	DKCDS	- ERROR CLOSING FILE & SAVING
11	DKCDS	- ERROR CLOSING FILE & DELETING
12	DKRBL	- READ INTO AN ACTIVE BUFFER - INTERNAL ERROR IN DH OR POLLUTED BUFFERS
13	DKASG	- CHARS IN FILE NAME MORE THAN ALLOWED
14	DKWAT	- INTERNAL ERROR - POLLUTED BT EMPTY - WAIT BIT IS ON
15	DKRBL	- I/O ERROR DURING READ
16	DKWBL	- WRITE FROM ACTIVE BUFFER
17	DKWBL	- I/O ERROR DURING WRITE
18	DKODS	- CAN'T PROCESS OLD FILE - NBSGNNBBLK INCONSISTENT

DATA-HANDLER PACKAGE ERROR NUMBERS (CONT'D)

ERROR NO.	ROUTINE	CONDITION
		WITH THOSE ASSIGNED VIA DKNIT
19	DKRNM	- OLD FILENAME DOES NOT EXIST
20	DKGET	- GET WORK THAN MAYGP CHARS
21	DKNSG	- CONTINUED BIT NOT SET EXPECT TO READ MORE SEGMENTS-LOGICAL INCONSISTENCY IN FILE
22	DKNSG	- CONTINUED BIT SET BUT NEXT LRID = OLOGICAL INCONSISTENCY IN FILE
23	DKSTO	- LR SEGMENT NOT IN PP (NON-ZERO SEG COUNT IN HEADER)
24	DKSTO	- PP HAS ZERO SEGS; ILLEGAL RECID
25	DKFND/DKGL2	- ACCESS OF A DELETED LR SEGMENT
26	DKGET	- ILLEGAL RECID (=0)
27	DKRNM	- INVALID OLD FILENAME
28	DKRNM	- INVALID NEW FILENAME
29	DKPUT	- L1.GT.LRLEN+1 ON UPDATE (EXTEND)
30	DKPUT	- FILE NOT OPEN
31	DKPUT	- L2.NE.0 ON ADD
32	DKPUT	- TRY TO PUT MORE THAN MAXGP CHARS
33	DKPUT	- INVALID FILENAME
34	DKPUT	- L1.LE.0 ON UPDATE
35	DKPUT	- ABS(L2).LT.L1 ON UPDATE
36	DKDEL	- LRID#0 FILE DOES NOT EXIST (RECORD DELETE)
37	DKDEL	- INVALID FILENAME
38	DKDEL	- PROTECTED FILE
39	DKDEL	- SHARED DS
40	DKOPN	- OPTION OUT OF RANGE
41	DKOPN	- ILLEGAL FILENAME
42	DKOPN	- OFT FULL
43	DKSOF	- FD FULL NO EXTEND YET
44	DKOPN	- PROTECTED FILE
45	DKOPN	- SHAPED DS
46	DKFIL	- ILLEGAL FILENAME
47	DKGLS	- ILLEGAL FILENAME
48	DKRNM	- NEW FILENAME ALREADY EXISTS
58	DKFIL	- CAN'T GET TO D.HAND TASK
68	UNUSE	- D
69	UNUSE	- D
70	DKGET	- L1.LT.0
71	DKGET	- L2.LT.0
72	DKGET	- L1.GT.LRLEN
73	DKXBM	- BIT MAP IS FULL. CANNOT BE EXTENDED.
74	DKXDS	- ALL BLOCKS IN DATASET USED.
75	LOCK/UNLOK	- NRSRCS NOT POSITIVE.
76	DKADT	- NO ROOM IN DATASET TABLE.

6 PROGRAM CONVERSION

The data-handler has been written to minimize the time required to convert the system from one computer to another. All conversion instructions are given as comment statements within the code of the following subroutines:

DKBEX
DKBIN
DKCDS
DKDDF
DKERR
DKGO1
DKJCL
DKMVC
DKNCS
DKNDS
DKNIT
DKODS
DKPAK
DKRBL
DKTRC
DKUPK
DKWAT
DKWBL
DKXDS
SNAP
TIMDAT

The programmer performing this conversion should follow the instructions given in Appendix C. Several subroutines have been written in IBM BAL to improve executive speeds. These routines must be rewritten if the data-handler is being converted for use on computer systems other than the IBM 360/370 series. The following is a list of the applicable routines:

DKBEX
DKBIN
DKDDF
DKJCL
DKMVC
DKNCS
DKPAK
DKUPK
TIMDAT

After each program has been rewritten in the new assembly language, the code should be added to the documentation tapes immediately after the previous version. The previous code should not be removed from the documentation tapes. This documentation system will allow the data-handler to be loaded in a relatively short time period onto a new computer system that is using a previously used assembler with minimal resources.

7 DOCUMENTATION TAPES

The documentation tapes provide a complete historic record of all computer programs ever written for the data-handler. All tapes are unlabeled, recorded at a density of 800 bits per inch (BPI), and blocked 800 characters per physical record. Each logical record is 80 characters in length.

The programs are listed in alphabetical order. Assembly programs are given in chronological order. The computer system and operating system version should be recorded in each assembly program in the first few comment cards.

All assembly programs are saved to allow conversion to a new computer assembly language with minimal resource expenditures. Each program is written as a separate tape file. The first tape file contains an index of all subsequent files. The format of this index is shown below:

DATA-HANDLER TAPE FILE INDEX

FILE NO.	PROGRAM NAME	SOURCE LANGUAGE	DATE WRITTEN	DATE MODIFIED	LAST
3--10 Right Justified	13--20 Left Justified	23--40 Left Justified	43--50 DD/MM/YY	53--60 DD/MM/YY	Card Columns Data Form

8 TEST ROUTINE

The last files on the documentation tapes contain a test routine that checks each application subroutine field. The correct results of this test routine are given on the tape for checking purposes.

9 CHANGES TO DATA-HANDLER PROGRAMS

As new functions are implemented into the data-handler, the following steps should be rigidly followed. Execution of these steps will insure that the changes are performed correctly and documented completely.

STEP ONE - Write a FORTRAN program, well documented with comment statements. Follow the standard documentation procedures below. If an existing program is being changed, change the FORTRAN program before changing the assembly programs.

Program Name: written as in the first line of the function or subroutine, with arguments.

Function: brief description of the function of the program.

Author: author's name, date of first writing.

Modifications: author, date and brief description of the reason for the change(s).

Language: programming language.

Calling Sequence: description of arguments and function results.

Routines Called: list of subroutines and functions called by the program.

Tasks or Modules: tasks or modules containing this routine.

Variables: a description of all variables used in the program. COMMON variables will generally be described in a "main" program, referenced in this program. The structure of arrays and meaning of variables should be fully described.

Program Logic: A detailed description of the algorithm(s) used and the flow of the program. The comments generating this portion of the documentation will normally be scattered throughout the program in complete English sentences.

STEP TWO - Change the last date modified on the data-handler tape file index.

STEP THREE - Change the FORTRAN compile listing on the documentation tapes.

STEP FOUR - (If required) - Write the assembly programs.

STEP FIVE - (If required) - Change the last date modified on the data-handler tape file index.

STEP SIX - (If required) - Change the assembler listing on the documentation tapes.

STEP SEVEN - Change the test routines to completely test the new feature.

STEP EIGHT - Execute the entire test routine package and replace the results on the documentation tapes.

STEP NINE - Revise this report as required. Document all revisions listed for this document at the beginning of the report.

STEP TEN - Revise all application programs as required by the change in the data-handler.

APPENDIX A

DATA HANDLER

DUMP EXAMPLE

```
****ERROR  30 IN DATA HANDLER PACKAGE.

TRACEBACK  ROUTINE  CALLED FROM ISN   REG.  14   REG.  15   REG.   0   REG.   1

           IHACL                0003   42215244   001B98A0   FFFFFDF4   00000000

           DKTHC                0046   42208DE4   00215180   FFFFFDF4   00000000

           DKEHR                0064   42212C74   00208C88   FFFFFDF4   00000000

           DKPUT                0003   420DC08E   002126F0   00000000   000DC084

           TEST1                0008   421407F0   000DC010   00000000   00000000

           EDISPC                      000239F0   001406F8   000D4F84   000D4F68

ENTRY POINT=  001406F8


2A9070     00000000  00000000  00000000  00000000  00000020  00000004  00000001  00000008  00000004  00000020  00000001  00000010
2A90A0     00006008  00000000  00000001  00000020  00000020  00000000  00000000  00000000  00000000  00000000  00000000  00000005
2A90D0     00000006  00000007  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
2A9100     00000000  00000001  00000000  00000000  FFFFFFFF  00000000  00000000  00000000  00000000  00000000  00000000  00000000
2A9130     00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
2A9160     00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
2A9190     00000000  00000000  00000000  000000C9  00002648  00000100  00000008  00000018  00000004  00000001  00000002  00000003
2A91C0     00000004  00000003  00000001  00000002  00000003  00005BD   0000001F  00000111  00000011  00000001  00000002  00000003
2A91F0     00000004  00000005  00000006  00000007  00000008  00000009  0000000A  0000000B  0000000F  00000010  00000011  000005BD
2A9220     000026CC  0000001D  0000001B  00000158  0000007D  00000009  00000001  00000005  00000006  00000007  00000008  00000009
2A9250     000000C8  0000000C  00000001  00000002  00000004  00000005  00000006  00000008  00000009  00000007  00000002  C9C2D440
2A9280     D6E24040  E5F1C640  C4C84040  0000001D  0000001E  0000001F  00000000  00000001  00000002  00000010  00000011  00000012
2A92B0     00000013  00000000  00000001  00000004  00000008  00000000  00000001  00000002  0000000A  0000000B  0000000C  00050001
2A92E0     C1F1E2E8  E2E3C5D4  00000100  00000001  00000000  00000100  00000014  00000015  00000069  00000001  001FFFFF  00000000
2A9310     00000001  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000014  00000000  00000000  00000000
2A9340     00000000  00000000  00000000  00000000  00000000  00000000  00007FFF  00000001  40404040  E2E8E2E3  C5D440E3  C1C2D3C5
2A9370     00000000  00000000  00000000  00000001  0000000C  00000000  00000000  0000001E

93

    NDS  <----FILNM-----> FULID    PREID    LRLEN  FSTAT ----FILNM----
      1      SYSTEM TABLE  00000000 00000000     0      1  40404040E2E8E2E3C5D440E3

  IPOOL  IPLEN  IOHER  IOHTR  IOKER  NMPSU   LFIN   LFIHU  NBLMX  NBSGN  NBBLK
    201   9800      0      0     30      1     32     32    256      8     24

***DATA SET TABLE
  IDSST    NDA   IELDS  ILDSN
    200     12      12      2

ENTRY NO.  1
   KDSLU <------KDSFN-------> KDSPR KDSPH KDSPL KDSSU KDSFD KDSFS KDSBI KDSWN    KDSWD
00050001 A1SYSTEM               256     1     0   256    20    21   105     1 001FFFFF

ENTRY NO.  2
   KDSLU <------KDSFN-------> KDSPR KDSPH KDSPL KDSSU KDSFD KDSFS KDSBI KDSWN    KDSWD
00040008 B1SYSTEM                 0     0     0     0     0     0     0     0 00000000
```

```
***OPEN FILE TABLE
  IOFST  NEOFT  IELOF
    344    125      9

ENTRY NO.   1
<----KOFFN----->        KOFFL       KOFLL     KOFNB     KOFLB       KOFFD  -----FILNM-----
ACS SYSTEM TABLE     00000000    00000000         3         0    00010101 C1C3E240E2E8E2E3C5D440E3

ENTRY NO.   2
<----KOFFN----->        KOFFL       KOFLL     KOFNB     KOFLB       KOFFD  -----FILNM-----
DIR SYSTEM TABLE     00000000    00000000         4         0    00020101 C4C9D940E2E8E2E3C5D440E3

***BUFFER TABLE
  IBTST   NEBT  IELBT  ILBTH  IBTLG
   1469     31    273     17   1469

ENTRY NO.  1
  KBTNS  KBTBL   KBTQ  KBTPP   KBTL  KBTTR  KBTCN  KBTDP  KBTNC     KBTSW  <----KBTFN----->     KBTFI     KBTPI  KBTLN
      0      0      0      0      0      0      0      0      0  00000000                    00000000  00000000      0

        0 00000    00000000    00000000    00000000    00000000        00000000    00000000    00000000    00000000

ENTRY NO. 29
  KBTNS  KBTBL   KBTQ  KBTPP   KBTL  KBTTR  KBTCN  KBTDP  KBTNC     KBTSW  <----KBTFN----->     KBTFI     KBTPI  KBTLN
      7      1      1   9167      0      0      0      0      0  00000001                    00000000  00000000      0

        0 00000    00020008    00000000    00000000    00000022        00000078    00000000    00000000    00000100
        8 00008    00000100    C4C84040    00000008    00000018        C9C2D440    D6E24040    E5F1C640    00000020
       16 00010    00000004    00000001    00000008    00000004        00000020    00000001    00000010    00000008
       24 00018    00000000    00000001    00000020    00000020        F1F57AF3    F67AF5F8    F1F060D6    C3E360F8
       32 00020    F0404040    00000000    00000002    00000000        00000001    000000D3    0000033C    00000000
       40 00028    00000000    00000001    00000000    00000000        00000000    00000000    00000000    00000003
       48 00030    00000000    00010003    00000000    00000003        00000000    00000000    00000000    00000000
       56 00038    00000000    00000000    00020000    00000000        00000004    00000000    00000000    00000000
       64 00040    00000000    00000000    00000000    00030000        00000000    00000005    00000000    00000000
       72 00048    00000000    00000000    00000000    00000000        00040000    00000000    00000006    00000000
       80 00050    00000000    00000000    00000000    00000000        00000000    00050000    00000000    00000007
       88 00058    00000000    00000000    00000000    00000000        00000000    00000000    00060000    00000000
       96 00060    00000008    00000000    00000000    00000000        00000000    00000000    00000000    00070000
      104 00068    00000000    00000009    00000000    00000000        00000000    00000000    00000000    00000000
      112 00070    00080000    00000000    0000000A    00000000        00000000    00000000    00000000    00000000
      120 00078    00000000    00090000    00000000    0000000B        00000000    00000000    00000000    00000000
      128 00080    00000000    00000000    000A0000    00000000        0000000C    00000000    00000000    00000000
      136 00088    00000000    00000000    00000000    000B0000        00000000    0000000D    00000000    00000000
      144 00090    00000000    00000000    00000000    00000000        000C0000    00000000    0000000E    00000000
      152 00098    00000000    00000000    00000000    00000000        00000000    000D0000    00000000    0000000F
      160 000A0    00000000    00000000    00000000    00000000        00000000    00000000    000E0000    00000000
      168 000A8    00000010    00000000    000F0000    00000000        00000000    00000000    00000000    000F0000
      176 000B0    00000000    00000011    00000000    00000000        00000000    00000000    00000000    00000000
      184 000B8    00100000    00000000    00000012    00000000        00000000    00000000    00000000    00000000
      192 000C0    00000000    00110000    00000000    00000013        00000000    00000000    00000000    00000000
      200 000C8    00000000    00000000    00120000    00000000        00000014    00000000    00000000    00000000
      208 000D0    00000000    00000000    00000000    00130000        00000000    00000015    00000000    00000000
      216 000D8    00000000    00000000    00000000    00000000        00140000    00000000    00000016    00000000
      224 000E0    00000000    00000000    00000000    00000000        00000000    00150000    00000000    00000017
      232 000E8    00000000    00000000    00000000    00000000        00000000    00000000    00160000    00000000
      240 000F0    00000000    00000000    00000000    00000000        00000000    00000000    00000000    00170000
      248 000F8    00000000    00000000    00000000    00000000        00000000    00000000    00000000    00000000
```

46

ENTRY NO. 31

KBTNS	KBTBL	KBTQ	KBTPP	KBTL	KBTTR	KBTCN	KBTDP	KBTHC	KBTSW	<----KBTFN----->	KBTFI	KBTPI	KBTLN
1	1	1	9713	0	0	0	0	0	00000001		00000000	00000000	0

0	00000	00020006	00000000	00000000	00000022	00000078	00000000	00000000	00000100
8	00008	00000100	C8C84040	00000008	00000018	C9C2D440	D6E24040	E5F1C640	00000020
16	00010	00000004	00000001	00000008	00000004	00000020	00000001	00000010	00000008
24	00018	00000000	00000001	00000020	00000020	F1F57AF3	F67AF4F9	F1F060D6	C3E360F8
32	00020	F0404040	00000000	00000002	00000000	00000001	000000D3	0000033C	00000000
40	00028	00000000	C1C3E240	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	00000003
48	00030	00000000	00010001	C4C9D940	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000
56	00038	00000004	00000000	00020001	E2D7C340	E2E8F2E3	C5D440E3	C1C2D3C5	00000000
64	00040	00000000	00000005	00000000	00030002	E4E2D940	E2E8E2E3	C5D440E3	C1C2D3C5
72	00048	00000000	00000000	00000006	00000000	00040001	C4D6C340	E2E8E2E3	C5D440E3
80	00050	C1C2D3C5	00000000	00000000	00000007	00000000	00050000	C6D6C640	E2E8E2E3
88	00058	C5D440E3	C1C2D3C5	00000000	00000000	00000008	00000000	00060000	C8C5C640
96	00060	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	00000009	00000000	00070000
104	00068	D7C7D540	E2E8F2E3	C5D440E3	C1C2D3C5	00000000	00000000	0000000A	00000000
112	00070	00080000	D7D9C640	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	0000000B
120	00078	00000000	00090000	D7D9D540	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000
128	00080	0000000C	00000000	000A0000	E3F3C640	E2E8E2E3	C5D440E3	C1C2D3C5	00000000
136	00088	00000000	0000000D	00000000	00080000	C2C4C440	E2E8E2E3	C5D440E3	C1C2D3C5
144	00090	00000000	00000000	0000000E	00000060	000C0003	C4C4E340	E2E8E2E3	C5D440E3
152	00098	C1C2D3C5	00000000	00000000	0000000F	00000000	000D0000	D2E8E6F1	E2E8E2E3
160	000A0	C5D440E3	C1C2D3C5	00000000	00000000	00000010	00000000	000E0000	D2E8E6F2
168	000A8	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	00000011	00000000	000F0000
176	000B0	D2E8E6F3	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	00000012	00000000
184	000B8	00100000	D2E8E6F4	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000	00000013
192	000C0	00000000	00110000	D2E8E6F5	E2E8E2E3	C5D440E3	C1C2D3C5	00000000	00000000
200	000C8	00000014	00000000	00120000	D2E8E6F6	E2E8E2E3	C5D440E3	C1C2D3C5	00000000
208	000D0	00000000	00000015	00000000	00130000	00000000	00000015	00000000	00000000
216	000D8	00000000	00000000	00000000	00000000	00140000	00000000	00000016	00000000
224	000E0	00000000	00000000	00000000	00000000	00000000	00150000	00000000	00000017
232	000E8	00000000	00000000	00000000	00000000	00000000	00000000	00160000	00000000
240	000F0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00170000
248	000F8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

***HT PRIORITY QUEUE

INTPQ	INTOP	INTHT
9932	29	27

ENTRIES=(FORWARD/BACKWARD) POINTERS

001E0002	00010003	00020004	00030005	00040006	00050007	00060008	00070009
0008000A	00090008	000A000C	000B0000	000C000E	000D000F	000E0010	000F0011
00100012	00110013	00120014	00130015	00140016	00150017	00160018	00170019
0018001A	0019001B	001A0000	001D001F	0000001C	001F0001	001C001E	

APPENDIX B : SUBROUTINE FUNCTION
B.1 APPLICATION PROGRAMS

NUMBER	SUBROUTINE NAME	FUNCTION
1.	DKDEL	ROUTINE TO DELETE FILE FILNM (IF RECID=0) OR LR DECID IF NOT 0 FILE DOES NOT HAVE TO BE OPEN BUT MUST EXIST FOR RECORD DELETE FILE DELETE OF NON-EXISTENT FILE IS NOT AN ERROR
2.	DKFIL	ROUTINE SET FSTAT (COMMON) TO INDICATE STATUS OF FILNM (COMMON)
3.	DKGET	GET LOGICAL RECORD RECID IN DATA SET NDS (COMMON)
4.	DKINS	INSERT LOGICAL RECORD INTO DATA SET NDS (COMMON) IN FILE FILNM (COMMON) AFTER LR RECID.
5.	DKLEN	ROUTINE TO GET LENGTH OF LR RECID IN DATA SET NDS (COMMON)
6.	DKLOS	ROUTINE TO CLOSE FILE FILNM (COMMON) IN DATA SET NDS (COMMON)
7.	DKNDS	GIVEN DATA SET NAME, RETURN DATA SET NUMBER
8.	DKNIT	INITIALIZES ALL DATA HANDLER VARIABLES
9.	DKOPN	OPEN A FILE ROUTINE.
10.	DKPUT	PUT LOGICAL RECORD INTO DATA SET NDS (COMMON) IN FILE FILNM (COM)
11.	DKRNM	ROUTINE TO RENAME FILE
12.	DKSET	ROUTINE TO RETURN INFORMATION ABOUT DATA SET NDS
13.	DKXIT	ROUTINE TO LOOP THRU ALL DATA HANDLER DATA SETS (NDA) AND CLOSE EM DOWN

B.2 FUNCTION PROGRAMS

SUBROUTINE NAME	FUNCTION
DKABP	ADD TO BUFFER PRIORITY: CURRENTLY KEEPS AN ORDERED LIST OF BT ENTRIES BY MOST RECENT USE. N IS IGNORED.
DKALO	ROUTINE TO TURN OFF THE PROTECT BIT FOR FILE FILNM IN DATA SET NDS. FILE NEED NOT BE OPEN
DKARG	CHECK VALIDITY OF NDS AND OPEN DATA SET IF NOT ALREADY OPEN
DKASG	ROUTINE TO ALLOW USER TO CONNECT A DATA SET NUMBER NSW TO A PARTICULAR LOGICAL UNIT NUMBERLUN AND FILE NAME FNAME.
DKBEX	INTEGER FUNCTION TO EXTRACT NUMBR BITS FROM SOURCE WORD
DKBIN	REAL FUNCTION TO INSERT THE LOW-ORDER NUMBR BITS FROM ISOR INTO RESLT AND RETURN IN REGISTER. RESIT IS NOT MODIFIED.
DKCBL	ROUTINE TO COMPACT A BLOCK AT BUFFERIBTADD CALCULATE NUMBER OF SU'S TO MOVE FROM POOL SU OTH DISPL IFROM TO POOL SU OTH DISPL ITO BY SETTING NO. SU'S IN BLOCK FROM DST AND NO. SU'S FREE AT END OF BLOCK (IN PR HEADER) MOVES SU'S ONE AT A TIME WITH DKPAK AND DKUPK AND UPDATES SU'S FREE AT END OF BLOCK. ASSUMES IFROM.GE.ITO EVENTUALLY MUST ALSO MODIFY FST
DKCDS	OP SYS FILE INTERFACE ROUTINE TO CLOSE A DATA SET
DKDD1	GET ALL MACHINE CHARS IN MACH (1-13)
DKDD2	DATA SET DUMP ROUTINES
DKDD3	DATA SET DUMP ROUTINES
DKDD4	NOT WRITTEN
DKDDF	ROUTINE FOR DYNAMIC DEFINE FILE STATEMENT IN OS FORTRAN ISSUES OPEN ON DDNAME FTNNF001
DKDDS	DUMP DATA SET ROUTINE TO PRODUCE A FORMATTED DUMP OF DATA SET NDS
DKDLF	ROUTINE TO DELETE ENTIRE FILNM (COMMON) IN DATA SET NDS (COMMON)
DKDL	RROUTINE TO DELETE LR (NSGNBL) IN FILE FILNM DATA SET NDS
DKDMP	DATA HANDLER

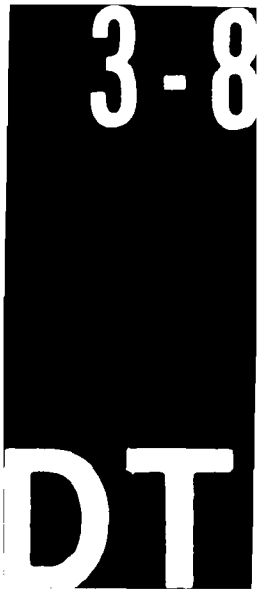
SUBROUTINE NAME	FUNCTION
DKDSG	ROUTINE TO DELETE THE SEGMENT WHOSE LR HEADER IS IN H IN BUFFER IBTADD ASSUMING KBTPP SET BY DKSTQ (VIA DKFND) DOES NOT PRESENTLY COMPACT THE BLOCK OR MODIFY FST
DKEBF	ROUTINE TO EMPTY THE BUFFER AT IBTAD WHOSE HEADER IS IN D CHECKS FOR PREVIOUS I/O COMPLETION WRITES AND WAITS IF WRITTEN INTO AND ZERO ENTRY AND D
DKERR	CKECKS ERROR HANDLING SWITCH AND EITHER PRINTS AND ABORTS OR RETURNS WITHOUT SAYING ANYTHING
DKFBF	FILL BUFFER ROUTINE: BT ENTRY AT IBTAD IS IN D GET BLOCK NBL OF DATA SET NSW IN AND WAIT FOR READ COMPLETION
DKFND	ROUTINE TO LOCATE BLOCK NBL SEGMENT NSG OF DATA SET NSW. SEARCHES BT'S FIRST·THEN READS INTO CORE RETURNS IBTAD=THE BT ENTRY DISPL WHERE PR IS DOES NOT SET MEMORY
DKGBF	GET A BUFFER ROUTINE: CHOOSES THE NEXT AVAILABLE BUFFER USING DKSBF ROUTINE SETS ITS PRIORITY HIGH WIHH DKABP AND EMPTIES IT WITH DKEBF. D AND BT ENTRY WILL BE ZEROED AND ALL IO FINISHED ON RETURN.
DKGBL	ROUTINE TO GET A BLOC5 FROM FREE SPACE TABLE OF DATA SET NDS WORD FROM BIT MAP STORED IN DST (IDSWD) NUMBER DST (IDSWN) BITS CORRESPOND TO BLOCKS NUMBER RT TO LEFT IN EACH WORD IF DST (IDSWN) = 0, NO WORD THERE. ALWAYS ASSUME WORD MUT BE REPLD USES INTERNAL ARRAY A TO STORE DST ENTRY
DKGBM	ROUTINE TO GET A BIT MAP WORD FOR NS WHICH HAS AT LEAST ONE BLOCK AVAILABLE (BIT=0) WITHIN THE BLOCKS INITIALIZED
DKLG1	ROUTINE TO GET THE SEGMENT OF THE LR (NSWNSEGNBLK) CONTAINING NCS INTO CORE BUFFER AND SETS D=BT HEADER H=LR HEADER SETS QPPP FOR BUFFER CONTAINING DESIRED SEGMENT ALSO SETS PREID FOLID AND LRLEN IN COMMON FROM EITHER LAST BT MEMORY IF SEGMENT FOUND THERE OR FROM FIRS AND LAST BUFFER EXCEPT FOR THE LOOP TO GET LAST SEGMENT
DKGLS	ROUTINE TO GET LAST SEGMENT OF LR (NSGNBL) IN NDS
DKGO	ROUTINE TO INITIALIZE THE DATA HANDLER BUILDS NECESSARY TABLES FROM ARRAY IN BLANK COMMON STARTING AT WORD ISTRT FOR ILEN WORS - ALSO SETS CONSTANTS AND INITIALIZES VARIABLES
DKGO1	SET MACHINE PARAMETERS IFR IBM 360/370 WITH OS

NUMBER	SUBROUTINE NAME	FUNCTION
	DKIDS	ROUTINE TO INITIALIZE THE RECORDS IN A NEW DATA SET IN PARTICULAR TO SET UP BLOCK 1 WITH TWO SEGMENTS:
	DKJCL	
	DKMFS	ROUTINE TO SEARCH THE FST AND LOCATE THE BLOCK WITH THE MOST FREE SPACE WHICH IS RETURNED IN NBL.
	DKMIC	ROUTINE TO SEARCH IN CORE BUFFER TABLES FOR ONE WITH DATA SET NDS FILENAME FILNM AND RETURN THE ONE WITH THE MOST FREE SPACE.
	DKMIR	ROUTINE FOR BUFFER MEMORY ID RESET TO ENSURE THAT WHEN A NEW LOGICAL RECORD (RID) IS ADDED OR INSERTED THE PRECEEDING (NSGPNBLP) AND FOLLOWING (NSGFNBLF) RECORDS (IF THEY APPEAR IN ANY BT MEMORY) HAVE THEIR FOLID AND PREID ENTRIES RESPECTIVELY EQUAL RID.
	DKMLR	ROUTINE FOR MEMORY LENGTH RESET IN BT ENTRIES
	DKMST	SETS BUFFER MEMORY PARAMETERS INTO D
	DKNCS	ROUTINE TO COMPUTE THE NUMBER OF CHARACTERS IN A LR SEGMENTS
	DKNSG	ROUTINE TO GET NEXT SEGMENT OF THE LR WHOSE HEADER IS IN H INTO A BUFFER SETS BT ADDR IN IBTAD BT ENTRY IN D.
	DKOUT	BRINGS ACTIVITY ON THIS DATA SET TO AN ORDERLY HALT CLOSE ANY OPEN DH FILES IN THIS DATA SET AND WAIT FOR ANY BUFFER I/O ACTIVITY.
	DKPA1	PUT ADD RECORD BUFFERING LOGIC - ACTION 1: LR WILL ENTIRELY FIT INTO IN-CORE BUFFER AT IBTADD
	DKPA2	PUT ADD RECORD BUFFERING LOGIC - ACTION 2: LR PLACED IN REMAINDER OF THE IN-CORE BUFFER AT IBTADD AND THEN IN ONE OR MORE NEW BLOCKS. THIS LATTER IS DONE VIA DKPA4.
	DKPA3	NOT WRITTEN YET
	DKPA4	PUT ADD RECORD BUFFERING LOGIC - ACTION 4: LR MUST BE PLACED IN ONE OR MORE NEW BLOCKS
	DKPA5	PROGRAM NOT WRITTEN YET
	DKPAK	ROUTINE TO PLACE ICT WORDS OF D (IN SU'S, NBU BITS) INTO POOL ARRAY
	DKPAR	ROUTINE TO DETERMINE THE PUT ADD RECORD BUFFERING RULE

NUMBER	SUBROUTINE NAME	FUNCTION
	DKPBL	PUT BLOCK NBL BACK IN BIT MAP OF DATA SET NDS SO IT CAN BE REUSED
	DKPBM	ROUTINE TO REPLACE THE NTH SU (IN WORD) IN THE BIT MAP OF NDS
	DKPMD(A)	ROUTINE TO UPDATE THE PTR IN THE LRID WRITTEN PRIOR TO CURRENT ONE IF IN CHAIN MODE
	DKPRO	ROUTINE TO SET PROTECT BIT FOR FILE FILNM IN DATA SET NDS
	DKPSG	PUT LR SEGMENT ROUTINE USED TO ADD A NEW SEGMENT AT END OF BUF
	DKPSH	ROUTINE TO REPLACE THE LR SEGMENT HEADER IN H INTO BUFFER
	DKRBL	READS BLOCK NBL FROM DATA SET NSW INTO BT ENTRY IBTAD(IN D)
	DKSBF	SELECT BUFFER ROUTINE
	DKSFD	SEARCH FILE DIRECTORY ROUTINE IN DS NSW
	DKSOF	SEARCH OPEN FILE TABLE ROUTINE FOR FILNM (COMMON) IN DATA SET NDS (COMMON) RETURNS IOFAD:
	DKSTQ	ROUTINE TO LOOP THRU BT ENTRY IBTAD AND LOCATE SEGMENT NSG
	DKTLR	ROUTINE TO TRUNCATE CURRENT SEGMENT AT CHAR NCE COMPACT THE BLOCK AND DELETE ANY FOLLOWING SEGMENTS.
	DKTRC	ROUTINE TO PROVIDE A TRACEBACK AND OTHER INFO WHEN ERROR CALLED BY: DKERR IBM OS VERSION H LEVEL OBJECT TIME SYSTEM
	DKUPK	ROUTINE TO PLACE ICT SU'S OF POOL ARRAY P BEGINNING AT ISTRT+1 INTO D (ONE SU PER WORD)
	DKWAT	WAIT ROUTINE: WAITS FOR BIT IBIT TO CLEAR IF SET IN BT STATUS WORD. ALLOWS ASYNCHRONOUS I/O OPERATIONS WITH THE PROPER OP SYS INTERFACE SUPPORT (DKRBL AND DKWBL).
	DKWBL	WRITES BLOCK NBL FROM BT ENTRY IBTAB(IN D) INTO DATA SET NSW
	DKWIN	ROUTINE TO SET THE 'WRITTEN INTO' BIT IN BT ENTRY IN D DOES NOT REPLACE D
	DKXBM	ROUTINE TO EXTEND BIT MAP BY ONE SEGMENT
	DKXDS	ROUTINE TO EXTEND DATA SET

NUMBER	SUBROUTINE NAME	FUNCTION
	DKXFD	EXTEND FILE DIRECTORY OF DATA SET NDS (COMMON) BY ONE BLOCK
	DKXLR	ROUTINE TO EXTEND THE LR (NSGNBL) WHOSE LAST SEGMENT IS IN BT ENTRY IBTAD,D AND WHOSE LR HEADER IS IN HEDLR
	XOR	
	DKODS	OPEN DATA SET TO HOST OP SYSTEMS FILE CONTROL SERVICES
	DKDMP	DUMPS DATA HANDLER COMMON AND POOL THIS ROUTINE IS A FUNCTION OF MACHINE CHARACTERISTICS SUCH AS CHARS PER WORD AND CHARS IN A DATA SET NAME
	GOT236	THIS ROUTINE IS ENTERED BY THE FORTRAN EXTENDED ERROR HANDLING ROUTINES
	BITEX	RIGHT JUSTIFY THE BITS TO BE RETURNED
	BITIN	THE RIGHTMOST NUMBR BITS OF SOURC ARE INSERTED INTO RESLT BEGINNING WITH BIT START OF RESLT. THE REMAINING BITS OF RESLT ARE NOT MODIFIED
	ICOPY	COPIES NUM CHARACTERS FROM SOURC (STARTING AT SFST) TO DEST (STARTING WITH DFST) ALL OTHER CHARACTERS OF DEST ARE UNCHANGED
	PACK	PACK NUM CHARACTERS INTO THE STRING DEST STARTING WITH CHARACTER L2 OF DEST. THE CHARACTERS ARE TAKEN FROM THE LEAST SIGNIFICANT CHARACTER OF THE ELEMENTS OF THE ARRAY SOURC STARTING WITH STANDARD UNIT L1.
	SNAP	
	SYSCM	INITIALIZE SYSTEM COMMON TO THE STANDARD VALUES REQUIRED BY THE PARTICULAR MACHINE IMPLEMENTATION
	DKMVC	MOVE NC CHARACTERS TO A(TO)+ITOFF
	DKBNF	ROUTINE TO INSERT AN INTEGER FORTRAN WORD INTO A REAL FORTRAN WORD
	DKBXF	ROUTINE TO EXTRACT AN INTEGER FORTRAN WORD FROM A REAL FORTRAN WORD.
	AND	
	DKCMP	

NUMBER	SUBROUTINE NAME	FUNCTION
	OR	
	SHIFT	
	TIMDAT	


3-8
DT